Scaling Up Large-scale Sparse Learning and Its Application to Medical Imaging

by

Qingyang Li

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2017 by the
Graduate Supervisory Committee:

Jieping Ye, Co-Chair
Guoliang Xue, Co-Chair
Jingrui He
Yalin Wang
Jing Li

ARIZONA STATE UNIVERSITY

May 2017

ABSTRACT

Large-scale ℓ_1-regularized loss minimization problems arise in high-dimensional applications such as compressed sensing and high-dimensional supervised learning, including classification and regression problems. In many applications, it remains challenging to apply the sparse learning model to large-scale problems that have massive data samples with high-dimensional features. One popular and promising strategy is to scaling up the optimization problem in parallel. Parallel solvers run multiple cores on a shared memory system or a distributed environment to speed up the computation, while the practical usage is limited by the huge dimension in the feature space and synchronization problems.

In this dissertation, I carry out the research along the direction with particular focuses on scaling up the optimization of sparse learning for supervised and unsupervised learning problems. For the supervised learning, I firstly propose an asynchronous parallel solver to optimize the large-scale sparse learning model in a multithreading environment. Moreover, I propose a distributed framework to conduct the learning process when the dataset is distributed stored among different machines. Then the proposed model is further extended to the studies of risk genetic factors for Alzheimer's Disease (AD) among different research institutions, integrating a group feature selection framework to rank the top risk SNPs for AD. For the unsupervised learning problem, I propose a highly efficient solver, termed Stochastic Coordinate Coding (SCC), scaling up the optimization of dictionary learning and sparse coding problems. The common issue for the medical imaging research is that the longitudinal features of patients among different time points are beneficial to study together. To further improve the dictionary learning model, I propose a multi-task dictionary learning method, learning the different task simultaneously and utilizing shared and individual dictionary to encode both consistent and changing imaging features.

DEDICATION

To my family

for their everlasting love

and supporting me

all the way.

ACKNOWLEDGMENTS

First and foremost, I would like to express my special appreciation thanks to my Ph.D. advisor Dr. Jieping Ye, for guiding and supporting me during the period when I pursue my Ph.D. at ASU. It is the most lucky things in my life to have Dr. Ye's guidance for my Ph.D.'s research. He is my research and life mentor to guide me to go through the long journey of the completion of my dissertation and subsequent Ph.D.. The joy and enthusiasm he has for the research was contagious and motivational for me, not only during tough times in the Ph.D. pursuit but also my future career.

I would also like to thank my Ph.D. co-advisor Dr. Guoliang Xue, for his tremendous effort and help for the accomplishment of this dissertation. Dr. Xue's optimization and game theory classes give me a lot of innovation and theoretical basis when I solve the real world machine learning and data mining problems. My sincere thanks also goes to my dissertation committee members: Dr. Yalin Wang, Dr. Jingrui He and Dr. Jing Li, for their guidance, insightful comments and understanding.

I am grateful to all the members of the big Yelab's family: Shuiwang Ji, Liang Sun, Jianhui Chen, Jiayu Zhou, Sen Yang, Yashu Liu, Shuo Xiang, Qian Sun, Tao Yang, Zhi Nie, Jie Wang, Zheng Wang, Pinghua Gong, Binbin Lin, Chao Zhang and Kefei Liu. It is a memorable time for the five years I spend at ASU due to many other friends. I would like to thank to: Zhiqin Zhu, Lin Chen, Yang Liu, Tianxiang Gao, Yinghua Wu, Jiliang Tang, Huiji Gao, Bing Li, Guanqiu Qi and Wu Li.

Last but not least, the completion of my Ph.D. could not be achieved without the support and encouragement from my family, especially my mother and my sister. The everlasting love from my family drives me to go through the twenty one years at school, a long journey from primary school, middle school, high school, university to the accomplishment of Ph.D. in the end. And thank Jie Zhang for her endless support, patience and encouragement since the day we met during this journey.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

Technological advances in data gathering have led to a rapid proliferation of big data in diverse areas such as the Internet, engineering, climate studies, cosmology, and medicine. Sparse regression models with ℓ_1-regularization are widely used to find the linear model of best fit. Many research efforts have been devoted to develop efficient solvers for the ℓ_1-regularized sparse learning models, such as the Lasso problem Tibshirani (1996). Recent technological innovations lead to huge data collections that keep growing rapidly. In order for this massive amount of data to make sense, new computational approaches are being introduced to let scientists and engineers analyze their data in a parallel and distributed manner. As a result, in many applications, running Lasso on huge-scale data sets usually exceeds the computing capacity of a single machine running single-threaded approaches. I wish to point out that for truly huge-scale problems it is absolutely necessary to *parallelize*. Parallelizing the learning process for the regression problem has recently drawn a lot of interest. This is in line with the rise and ever increasing availability of high performance computing systems built around multi-core processors, GPU-accelerators and computer clusters, the success of which is rooted in massive parallelization.

Scaling up the learning process in parallel can be divided into two categories: parallel solvers and distributed solvers. Parallel solvers focus on shared-memory architectures in a multithreading environment, and in particular most of the solvers use the multithreading programming model, such as OpenMP. For distributed solvers, communication among the computing nodes is based on either the shared memory or distributed memory. In this dissertation, I focus on algorithms using the distributed

memory, as they can often handle much larger data sets. The infrastructure to deploy and implement the distributed solvers are distributed platform, such as Hadoop, Spark and MPI. In this study, I carry out research along these directions aiming to propose novel parallel and distributed solvers to scale up the learning process of large-scale machine learning problems.

Coordinate descent (CD) is one of the most successful classes of algorithms in the big data optimization domain. Broadly speaking, CD is based on the strategy of updating a single coordinate (or a single block of coordinates) of the vector of variables at each iteration. This often drastically reduces memory requirements as well as the arithmetic complexity of a single iteration, making the methods easily implementable and scalable. Most of the proposed parallel algorithms are based on Stochastic Coordinate Descent (SCD) Shalev-Shwartz and Tewari (2011) to accelerate the whole learning process. Many existing parallel solvers, like Shotgun Kyrola *et al.* (2011), Parallel Block Coordinate Descent (PBCD) Richtárik and Takáč (2016) and Thread-Greedy Scherrer *et al.* (2012b,a), employ multiple-threaded computing by utilizing multiple cores on a shared memory system. However, the curse-of-dimensionality is still a great challenge for large-scale problems. High-dimensional data in feature space means more time spent in the optimization and data synchronization in the multithreading environment.

To address this issue, *screening* is one of highly efficient approaches to solve the high-dimensional problem. Screening pre-identifies inactive features that have zero components in the solution and remove them from the optimization. As a result, I can solve the regression problem on the reduced feature matrix, leading to substantial savings in terms of computation and memory usage. The idea of screening has achieved great success in a large class of ℓ_1-regularized problems Ghaoui *et al.* (2012); Tibshirani *et al.* (2012a); Wang *et al.* (2013a,b); Wang and Ye (2014); Wang *et al.*

(2014); Wang and Ye (2015b,a), such as Lasso regression, Logistic regression, elastic net, multi-task feature learning (MTFL) and more general convex problems. Parallel screening is a promising strategy to solve the high-dimensional problem in big data optimization. However, the idea of using screening in the multithreading and distributed environment has not been investigated, since it is challenging to integrate screening rules with parallel solvers.

For the distributed solvers, a well-known approach is based on the Alternating Direction Method of Multipliers (ADMM) Boyd *et al.* (2011), also known as an operator splitting scheme. For many convex problems including those in sparse optimization, it gives rise to their parallel and distributed algorithms Mota *et al.* (2012). However, distributed ADMM does not scale well; given a fixed amount of data, distributing the data and ADMM computation to more nodes do not reduce its running time, because its number of iterations increases with the number of distributed data blocks, the time saved due to a smaller block size being offset by the increased number of iterations. In this dissertation, I propose a distributed framework to resolve the above issues.

Another issue in the distributed solvers is data collection and data storage. Recent advances in data collection technologies Tsai *et al.* (2013a,b) have made it possible to collect a large amount of data for many application domains. Very often, these data come from multiple sources. However, it is not a good idea for distributed solvers to collect data from multiple sources and store it in every machine. For instance, in the study of Alzheimer's Disease (AD), different research institutions share genomic data publicly under certain conditions, but more commonly, each participating institution may be required to keep their genomic data private, so collecting all data together may not be feasible. It is necessary to develop a distributed learning approach to conduct the learning process without compromising data privacy for each participating institutions.

A key challenge in applying sparse learning to biological and medical imaging problems is that the available labeled training samples are very limited. It is necessary to conduct the unsupervised learning to generate the features for the biomedical problems. Dictionary learning and sparse coding Lee *et al.* (2007); Mairal *et al.* (2009) is one of the highly efficient unsupervised sparse learning methods, using a small number of basis vectors to represent local features effectively and concisely to help with image content analysis and other computer vision problems. Sparse coding concerns the problem of reconstructing data vectors using sparse linear combinations of basis vectors Olshausen and Field (1996); Chen *et al.* (2001); Donoho and Elad (2003). It has become extremely popular for learning the dictionary and extracting features from images in the last decade. Sparse coding has been applied in many fields including audio processing Smith and Lewicki (2006), text mining Balakrishnan and Madigan (2008); Zhang *et al.* (2014) and image recognition Szlam *et al.* (2012). Different from traditional feature extraction methods like principal component analysis and its variants, sparse coding learns non-orthogonal and overcomplete dictionaries which have more flexibility to represent the data. Sparse coding can also model inhibition between the bases by sparsifying their activations. Similar properties have been observed in biological neurons, thus making sparse coding a plausible model of the visual cortex Olshausen and Field (1997, 2004). Recently, several work based on stochastic gradient descent and online learning has been proposed. Mairal *et al.* (2009) proposed an online dictionary learning algorithm which updates the dictionary for each incoming data point. It is expected that the learnt dictionary will converge faster in the online setting. However, even when the dictionary has been learned, one has to further learn the sparse code, which is also computationally expensive especially for large-scale data sets. In this study, I propose a novel approach for efficiently scaling up the optimization of the dictionary learning and sparse coding problem.

Algorithmic image-based diagnosis and prognosis of neurodegenerative diseases on longitudinal data has drawn great interest from computer vision researchers. The common issue for the medical imaging research is that the longitudinal features of patients among different time points or features from different region of interests (ROIs) will always be beneficial to study together. Recently, Multi-Task Learning (MTL) has been successfully explored for regression with different time slots. The idea of multi-task learning is to utilize the intrinsic relationships among multiple related tasks in order to improve the prediction performance. One way of modeling multi-task relationship is to assume all tasks are related and the task models are connected to each other Evgeniou *et al.* (2005), or the tasks are clustered into groups Zhou *et al.* (2012). Alternatively, one can assume that tasks share a common subspace Chen *et al.* (2009), or a common set of features Argyriou *et al.* (2008). Recently, Maurer *et al.* Maurer *et al.* (2013) proposed a sparse coding model for MTL problems based on the generative methods. In this study, I proposed a novel unsupervised multi-source dictionary learning method to learn the different tasks simultaneously which utilizes shared and individual dictionaries to encode both consistent and individual imaging features for longitudinal image data analysis.

The rest of this dissertation is organized as follows. In Chapter 2, I propose a parallel solver, termed Asynchronous Grouped Coordinate Descent (AGCD), to scale up the optimization of sparse regression problem in a multithreading environment. In Chapter 3, I propose a distributed framework to optimize the regression model in a distributed manner and integrate it with the studies of risk genetics factors of AD. In Chapter 4, I show that by integrating a distributed group feature selection framework, more associated risk SNPs are detected by the proposed method. In Chapter 5, I propose a unsupervised efficient method, termed Stochastic Coordinate Coding (SCC), scaling up the dictionary learning and sparse coding problems, applying

on feature extraction for image classification and other computer vision application. In Chapter 6, the proposed SCC is extended to a multi-task learning framework to deal with the longitudinal data sets since the biomedical subject might have different representations at different time points or multiple brain ROIs. I summarize the major contributions of this dissertation and discuss some future research directions in Chapter 7.

Chapter 2

SCALING UP SPARSE REGRESSION MODELS IN PARALLEL

2.1 Introduction

In many real word applications, we encounter very high-dimensional data such as images, texts, and genomic data when optimizing the machine learning problems. One promising approach is to parallelize the solver so that the optimization process can be accelerated by utilizing multiples processors or computation nodes. Most of the published parallel solvers are based on parallelizing SCD to speedup the optimization process. The updating strategy of SCD is to randomly select one coordinate to update in each iteration. Parallelizing SCD allows multi-processors to update the coordinates concurrently without synchronization. Although it might result in the divergence of objective function, parallel solvers can achieve dramatic speedup when optimizing the regression problem in a very high-dimensional feature space. It is shown in Kyrola *et al.* (2011) that the dimension of feature space for parallel solvers should be no less than $\frac{P}{2\rho}$ when there are P threads, where ρ denotes the spectral radius of A^TA, and A is the feature matrix. When we integrate screening methods with the state-of-the-art parallel solvers such as Shotgun, PBCD and Asynchronous Stochastic Coordinate Descent (ASYSCD) Liu *et al.* (2014), it can result in the divergence of the objective function since the feature matrix is shrunk to a matrix with a small feature space after applying screening rules on it. Although we can reduce the number of threads to guarantee the convergence, it has a negative effect on the scalability of the parallel method. Since previous parallel solvers cannot satisfy the constraint between the number of threads and feature space, it is essential to develop a parallel solver to

optimize the problem in the reduced feature data matrix.

2.2 Problem Formulation

In this study, I consider the following ℓ_1-regularized minimization problem:

$$\min_{x \in \mathbb{R}^N} F(x) = \frac{1}{2}\|Ax - y\|_2^2 + \lambda \|x\|_1, \tag{2.1}$$

where A is the design matrix and $A \in \mathbb{R}^{M \times N}$, $y \in \mathbb{R}^M$ is the response vector and x is the sparse model we need to learn. λ is the regularization parameter and $\lambda > 0$.

In this study, I propose a parallel framework to solve the Lasso regression problem on large-scale datasets with huge dimensional feature space. I parallelize screening rules by partitioning the sample and feature space to accelerate the screening process. I propose two parallel safe screening rules: Parallel Strong Rule (PSR) and Parallel Dual Polytope Projection (PDPP). To optimize the regression problem in parallel on the reduced feature matrix, I propose an Asynchronous Grouped Coordinate Descent method (AGCD) to solve the problem of small feature space in the optimization after employing screening rules. In AGCD, I introduce competition strategy to select the candidate coordinates that minimize the objective function with the maximum descent of function value. If the selected coordinate wins the competition in a group of candidates, that coordinate will be updated at this iteration, otherwise the update terminates. The main idea of AGCD is to reduce the frequency to update coordinates and select the most important candidates to update, allowing the solver to converge in a small feature space. It is different with the random selection strategy in most of the parallel solvers.

Screening

Existing screening methods can be divided into two categories: *Safe Screening Rules* and *Heuristic Screening Rules*. Safe screening rules guarantee that the predicted inactive features have zero coefficients in the solution. In other words, discarding the inactive features in safe screening rules does not sacrifice the accuracy of optimization since the corresponding positions in the solution vector are zero in the ground truth. SAFE Ghaoui *et al.* (2012) is a safe screening method that estimates the dual optimal solution of Lasso. Strong rules Tibshirani *et al.* (2012a) are another efficient screening methods based on heuristic screening rules. In most of the cases, strong rules discard more features than SAFE, leading to a substantial speedup. However, strong rules cannot guarantee that discarded features have zero components in the solution. To avoid the incorrectly discarded cases, Tibshirani *et al.* (2012a) proposed a method to check the KKT conditions, ensuring the correctness of screening results. To optimize the problem along a sequence of parameter values, Enhanced Dual Polytope Projection (EDPP) Wang *et al.* (2013a) is an efficient and effective safe screening method and achieves significant speedup for the Lasso problem.

Parallel Solvers

After we get the reduced feature matrix from screening rules, we can apply different solvers to optimize it, such as Stochastic Gradient Descent (SGD) Zhang (2004), FISTA Beck and Teboulle (2009), ADMM Boyd *et al.* (2011) and SCD Shalev-Shwartz and Tewari (2011); Nesterov (2012); Richtárik and Takáč (2014). When we consider solvers in a multithread environment, a lot of parallel solvers were proposed based on the SCD. Shotgun Kyrola *et al.* (2011) is a parallel coordinate descent method which allows multiple processors to update the coordinates concurrently. PBCD Richtárik

and Takáč (2016) described a method for the convex composite optimization problem. In this method, all the processors update randomly selected coordinates or blocks synchronously at each iteration. The method in Li and Osher (2009); Scherrer *et al.* (2012b,a) are based on the greedy coordinate descent. Recently, asynchronous parallel methods are proposed to accelerate the updating process. Asynchronous Stochastic Coordinate Descent (ASYSCD) Liu *et al.* (2014) proved the linear convergence of asynchronous SCD solver under the essential strong convexity condition. PASSCoDe Hsieh *et al.* (2015) is an asynchronous Stochastic Dual Coordinate Descent(SDCD) method for solving the dual problem. Parallel SDCA Tran *et al.* (2015) is an asynchronous parallel solver based on the Stochastic Dual Coordinate Ascent (SDCA) method. Both PassCoDe and Parallel SDCA focus on the ℓ_2-regularized models.

2.3 Proposed Framework

In this section, I present the proposed parallel framework based on a shared memory model with multiple processors. My parallel framework is composed of two main procedures:

1. Identify the inactive features by the parallel screening rules and remove inactive features from optimization.

2. Solve the Lasso regression on the reduced feature matrix in parallel.

In step one, I parallelize screening rules to identify and discard inactive features, significantly accelerating the whole learning process. I propose two parallel screening rules: PSR and PDPP.

In step two, I propose an asynchronous parallel solver AGCD to solve the Lasso regression on the reduced data matrix in parallel.

The dual problem of Lasso (2.1) can be formulated as the following equation:

$$\sup_{\theta} \left\{ \frac{1}{2}||y||_2^2 - \frac{\lambda^2}{2}||\theta - \frac{y}{\lambda}||_2^2 : \left|[A]_j^T \theta\right| \leq 1, j = 1, 2, ..., p \right\}, \tag{2.2}$$

where θ is the dual variable and $[A]_j$ denotes the jth column of A. Let $\theta^*(\lambda)$ be the optimal solution of problem (2.2) and $x^*(\lambda)$ denotes the optimal solution of problem (2.1). The Karush–Kuhn–Tucker (KKT) conditions are given by:

$$y = Ax^*(\lambda) + \lambda\theta^*(\lambda), \tag{2.3}$$

$$[A]_j^T \theta^*(\lambda) \in \begin{cases} \text{sign}([x^*(\lambda)]_j), & \text{If } [x^*(\lambda)]_j \neq 0, \\ [-1, 1], & \text{If } [x^*(\lambda)]_j = 0, \end{cases} \tag{2.4}$$

where $[x^*(\lambda)]_k$ denotes the kth component of $x^*(\lambda)$. In view of the KKT condition in equation (2.4), the following rule holds:

$$\left|[A]_j^T \theta^*(\lambda)\right| < 1 \Rightarrow [x^*(\lambda)]_j = 0 \Rightarrow x_j \text{ is an inactive feature.} \tag{2.5}$$

The inactive features have zero components in the optimal solution vector $x^*(\lambda)$ so that we can remove them from the optimization without sacrificing the accuracy of the optimal value in the objective function (2.1). We call this kind of screening methods as *Safe Screening Rules*. SAFE Ghaoui *et al.* (2012) is one of highly efficient safe screening methods. In SAFE, the jth entry of $x^*(\lambda)$ is discarded when

$$\left|[A]_j^T y\right| < \lambda - ||[A]_j||_2||y||_2 \frac{\lambda_{\max} - \lambda}{\lambda_{\max}}, \tag{2.6}$$

where $\lambda_{\max} = \max_j \left|[A]_j^T y\right|$. As a result, the optimization can be performed on the reduced data matrix $\widetilde{A}$ and the original problem (2.1) can be reformulated as:

$$\min_{\widetilde{x}} \frac{1}{2}||\widetilde{A}\widetilde{x} - y||_2^2 + \lambda||\widetilde{x}||_1 : \widetilde{x} \in \mathbb{R}^{\widetilde{p}} \text{ and } \widetilde{A} \in \mathbb{R}^{n \times \widetilde{p}}, \tag{2.7}$$

where $\widetilde{p}$ is the number of remaining features after employing safe screening rules. The optimization is performed on a reduced feature matrix, accelerating the whole learning process significantly.

Strong rules Tibshirani *et al.* (2012a) are another efficient screening methods based on heuristic screening method. In strong rules, the ith feature will be discarded when satisfies the following equation:

$$|A_i^T y| < 2\lambda - \lambda_{\max}. \tag{2.8}$$

The calculation of $\lambda_{\max}$ follows the same way in SAFE.

2.4 Parallel Screening Rules

Parallel SAFE and Parallel Strong Rules

For large-scale problems, it is necessary to parallelize the learning process. To speedup the learning process, I parallelize screening rules in a multithreading environment. Suppose there are P processors, I partition the feature space into P parts. The jth processor holds a subset S_j of feature space where $S_j \subseteq S$ and $S = \{1, 2, ..., N\}$. To average the performance of parallel solvers, each thread holds N/P coordinates and the partition is non-overlapped. I summarize the Parallel SAFE rule (PSAFE) in algorithm (1).

At the beginning, every processor generates the index set S_j and $S_j \in \mathbb{R}^{N/P}$. Let us define some notations here. E is a vector and $E \in \mathbb{R}^{N/P}$. $[W]_{S_j}$ indicates the collection of ωth element in W where $\omega \in S_j$ if W is a vector. When W represents a matrix, $[W]_{S_j}$ denotes the collection of ωth column of W and $\omega \in S_j$. Since $\lambda_{\max} = \max_i |A_i^T y|$, I first need to compute $E = A^T y$ firstly. To achieve this on P processors, I partition the computation into P parts. Every processor performs $[E]_{S_j} = [A]_{S_j}^T y$ in parallel. The time complexity is reduced from O(MN) to O(MN/P) since no

Algorithm 1 Parallel SAFE rule (PSAFE)

Require: A, y and λ.

Ensure: $\widetilde{A}$ and the selected index set I.

Initialize: $I = \mathbf{0} \in \mathbb{R}^N$.

In parallel on P processors.

Generate the index set S_j for the jth processor.

Compute the λ_{max}: $[E]_{S_j} = [A]_{S_j}^T y$, $\lambda_{\max} = \|E\|_\infty$.

Get the norm of response y: $\sigma = \|y\|_2$.

for each element i in the set of S_j **do**

Get the norm of the ith column of A: $\tau = \|A_i\|_2$.

if $|A_i^T y| \geq \lambda - \tau * \sigma * \frac{\lambda_{\max} - \lambda}{\lambda_{\max}}$ **then**

$I_i =$ **true**, select ith column of A into $\widetilde{A}$.

end if

end for

synchronization is needed between processors. E is stored as a global variable and can be accessed by all the processors after updated. Then I get $\lambda_{\max}$ by $\|E\|_\infty$. From the 6th line to the 11th line in algorithm 1, every processor performs screening rules on its own index set S_j to select the active features. Since A and b are global variables, all the processors are able to calculate σ and τ in parallel without synchronization. In the end, I get the selected index set I and reduced feature matrix $\widetilde{A}$. Suppose I has $\widetilde{N}$ elements of true values, the dimension of $\widetilde{A}$ is $\mathbb{R}^{M \times \widetilde{N}}$. The original optimization problem (2.1) can be reformulated as:

$$\min_{\widetilde{x}} \frac{1}{2}\|\widetilde{A}\widetilde{x} - y\|_2^2 + \lambda\|\widetilde{x}\|_1 : \widetilde{x} \in \mathbb{R}^{\widetilde{N}}. \tag{2.9}$$

After I obtain the solution vector $\widetilde{x}^*$ for problem (2.9), I am able to cover the x^* by I and $\widetilde{x}^*$.

The implementation of PSR in parallel follows the same way of PSAFE. The only difference between PSR and PSAFE is that the computation of $\|A_i\|_2$ and $\|y\|_2$ in equation (2.6) is not needed in PSR. I employ the same partition strategy and parallel technique to parallelize the strong rules in equation (2.8). Therefore, I skip the details of implementation for brevity.

Parallel Dual Polytope Projection

In many machine learning applications, the optimal value of the regularization parameter λ is unknown. To tune the value of λ, commonly used methods such as cross validation needs to solve the Lasso problem along a sequence of parameter values $\lambda_0 > \lambda_1 > ... > \lambda_\kappa$, which can be very time-consuming. A sequential version of strong rules was proposed in Tibshirani *et al.* (2012a) by utilizing the information of optimal solutions in the previous parameter. Suppose I have already obtained the solution vector $x(\lambda_{k-1})^*$ at λ_{k-1} where the integer $k \in [1, \kappa]$, the sequential strong rule rejects the ith feature at λ_k when the following rule holds:

$$|A_i^T(y - Ax(\lambda_{k-1})^*| < 2\lambda_k - \lambda_{k-1}. \tag{2.10}$$

Although the sequential strong rule is able to predict a large proportion of inactive features, it might mistakenly discard active features that have nonzero components in the solution. Therefore, I need to check the KKT conditions to guarantee the correctness of the predicted results.

EDPP Wang *et al.* (2013a) is a highly efficient safe screening method that estimates the dual problem and geometric properties of Lasso regression, achieving significant speedups for real-world applications. The implementation details of EDPP is available on the GitHub [1] . I omit the introduction of EPDD for brevity. Based on

[1] http://dpc-screening.github.io/lasso.html

the partition strategy and parallel technology employed on PSAFE and PSR, I propose a parallel safe screening rules, known as the Parallel Dual Polytope Projection (PDPP), to quickly identify and discard inactive features parallelly in a sequence of parameters.

To parallelize the screening rules, I need to partition both the feature space and sample space. In Chapter 2.5.1, this is done in the feature space, and I follow a similar way to partition it in the sample space. Before introducing the details about the proposed algorithm, I first introduce notations in the study. As I discussed in Chapter 2.5.1, I use $[W]_{S_j}$ to indicate the collection of elements of W in the index set S_j if W denotes a vector. When W is a matrix, I use the $[W]_{S_j}$ to represent the corresponding columns of W in the index set S_j. I use the same notations in PDPP. Suppose there are P processors, I partition the sample space into P parts. The jth processor holds an index set T_j of sample space, where $T_j \subseteq T$ and $T = \{1, 2, ..., M\}$. Every subset T_j has M/P elements and there is no overlap among them. When W denotes a vector, $\{W\}_{T_j}$ indicates the collection of every ωth elements from W in the index set T_j where $\omega \in T_j$. When W is a data matrix, $\{W\}_{T_j}$ denotes the collection of every ωth rows in W where $\omega \in T_j$. To take $\{A\}_{T_j}$ as an example, I extract columns of A in the index set T_j to construct it. So the dimension of $\{A\}_{T_j}$ is $\mathbb{R}^{\frac{M}{P} \times N}$. I summarize the PDPP method in algorithm 2.

In PDPP, all the P processors perform the computation in parallel. Firstly, the jth processor generates the corresponding index set S_j and T_j by the method I discussed previously. Then I follow the same way in PSAFE and PSR to calculate the $\lambda_{\max}$ in parallel. The dimensions of $\theta(\lambda)$, $v_1(\lambda_0)$, $v_2(\lambda, \lambda_0)$, $v_2^{\perp}(\lambda, \lambda_0)$ are $\mathbb{R}^M$. In PDPP, I employ partition strategy in sample space on these variables: $\{\theta(\lambda)\}_{T_j}$, $\{v_1(\lambda_0)\}_{T_j}$, $\{v_2(\lambda, \lambda_0)\}_{T_j}$ and $\{v_2^{\perp}(\lambda, \lambda_0)\}_{T_j}$. As a result, the computation of these variables is performed in parallel. From line 19 to line 25 in algorithm 2, I employ PDPP on a

Algorithm 2 Parallel Dual Polytope Projection (PDPP)

In parallel on P processors

Generate the S_j and T_j for the jth processor.

Compute the λ_{max}: $[E]_{S_j} = [A]_{S_j}^T y$, $\lambda_{\max} = \|E\|_\infty$.

$\phi = argmax_i |E|$, $v = A_\phi$, v is the ϕth column of A.

Let $\lambda_0 \in (0, \lambda_{\max}]$ and $\lambda \in (0, \lambda_0]$.

if $\lambda = \lambda_{\max}$ **then**

$\{\theta(\lambda)\}_{T_j} = \frac{\{y\}_{T_j}}{\lambda_{\max}}$.

else

$\{\theta(\lambda)\}_{T_j} = \frac{\{y\}_{T_j} - \{A\}_{T_j} x(\lambda)^*}{\lambda}$.

end if

if $\lambda_0 = \lambda_{\max}$ **then**

$\{v_1(\lambda_0)\}_{T_j} = sign(v^T y)\{v\}_{T_j}$.

else

$\{v_1(\lambda_0)\}_{T_j} = \frac{\{y\}_{T_j}}{\lambda_0} - \{\theta(\lambda_0)\}_{T_j}$.

end if

$\{v_2(\lambda, \lambda_0)\}_{T_j} = \frac{\{y\}_{T_j}}{\lambda} - \{\theta(\lambda_0)\}_{T_j}$.

$\alpha = \frac{<v_1(\lambda_0), v_2(\lambda, \lambda_0)>}{\|v_1(\lambda_0)\|_2^2}$.

$\{v_2^\perp(\lambda, \lambda_0)\}_{T_j} = \{v_2(\lambda, \lambda_0)\}_{T_j} - \alpha\{v_1(\lambda_0)\}_{T_j}$.

Given $\lambda_{\max} = \lambda_0 > ... > \lambda_\kappa$, for $k \in [1, \kappa]$, I make a prediction of screening on λ_k

if $x(\lambda_{k-1})^*$ is known:

$[w]_{S_j} = [A]_{S_j}^T (\theta(\lambda_{k-1}) + \frac{1}{2} v_2^\perp(\lambda_k, \lambda_{k-1}))$

for every element i in the set of S_j **do**

if $w_i < 1 - \frac{1}{2}\|v_2^\perp(\lambda_k, \lambda_{k-1})\|_2 \|A_i\|_2$ **then**

Discard the i th column from A.

end if

end for

sequence of parameter values: $\lambda_{\max} = \lambda_0 > ... > \lambda_\kappa$. When performing the screening rule on λ_k and $k \in [1, \kappa]$, I need to compute w firstly, where $w \in \mathbb{R}^N$. I perform the computation of w based on the partition strategy in the feature space:

$$[w]_{S_j} = [A]_{S_j}^T(\theta(\lambda_{k-1}) + \frac{1}{2}v_2^\perp(\lambda_k, \lambda_{k-1})). \tag{2.11}$$

Finally, for each element i in the index set S_j, I will identify the ith entry of $x(\lambda_k)^*$ to be zero if the following rule holds:

$$w_i \geq 1 - \frac{1}{2}\|v_2^\perp(\lambda_k, \lambda_{k-1})\|_2\|A_i\|_2. \tag{2.12}$$

The calculation of $\|v_2^\perp(\lambda_k, \lambda_{k-1})\|_2$ and $\|A_i\|_2$ in (2.12) is similar to the calculation of $\|y\|_2$ and $\|A_i\|_2$ in algorithm 1.

Overall, the time complexity of PDPP is O(MN/P). Regardless of the calculation and updating on the vector variables, the calculation of these variables is dominant: w, $\{\theta(\lambda)\}_{T_j}$ and $\|A_i\|_2$ where $i \in S_j$. The calculation of all these variables can be parallelized by the partition strategy in PDPP without synchronization among processors.

2.5 Asynchronous Parallel Solver

Asynchronous Grouped Coordinate Descent

To address the challenge I discussed in Chapter 2.3.2, I propose a novel parallel solver, called Asynchronous Grouped Coordinate Descent (AGCD), to solve the Lasso regression on the reduced feature matrix. Rather than randomly selecting coordinates or blocks to update asynchronously among threads, AGCD adopts a grouped selection strategy; that is, chooses the candidate that minimizes the objective function with the most descent to update among a group of coordinates. The details of AGCD are given in algorithm 3.

In AGCD, there are two global variables d and R to store where $d \in \mathbb{R}^{\widetilde{N}}$ and $R \in \mathbb{R}^{M}$. I initialize d to be zero and R to be $-y$. In each iteration, every processor randomly picks up a coordinate i from $\{1, 2, ..., \widetilde{N}\}$ to estimate and update. The calculation of the gradient for the ith coordinate, which is denotes as $g(\widetilde{x})_i$, can be written as:

$$g(\widetilde{x})_i = \widetilde{A}_i^T(\widetilde{A}\widetilde{x} - y). \tag{2.13}$$

To make it more efficient, the calculation of (2.13) can be decomposed into the following steps:

Step1: Calculate the gradient: $g(\widetilde{x})_i = \widetilde{A}_i^T R$ and get $\Delta\widetilde{x}_i$.

Step2: Update R : $R = R + \Delta\widetilde{x}_i\widetilde{A}_i^T$.

Since R is initialized as $-y$, R stores the information of $\widetilde{A}_i^T(\widetilde{A}x - y)$ by following the above updating rules. To calculate $\Delta\widetilde{x}_i$, I apply soft thresholding function Shalev-Shwartz and Tewari (2011) to get the proximal gradient for $\widetilde{x}_i$. The definition of soft thresholding operator Γ is given by :

$$\Gamma_\varphi(x) = sign(x)(|x| - \varphi). \tag{2.14}$$

In algorithm 3, L denotes the Lipschitz constant. For SCD Shalev-Shwartz and Tewari (2011); Nesterov (2012); Richtárik and Takáč (2014), the Lipschitz constant is set to be $\|A_i\|_2^2$ when updating the ith coordinate. Since SCD randomly picks only one coordinate to update, the problem has a closed-form solution in each iteration. When considering a multithreading environment, the way to calculate L is different. PBCD Richtárik and Takáč (2016) employs Expectation Maximization (EM) to get an approximation model on L but it depends on the sparsity of the feature matrix. In this study, I employ the same method in Liu *et al.* (2014) to get the Lipschitz constant from the Hassian matrix.

Algorithm 3 Asynchronous Grouped Coordinate Descent (AGCD)

Require: $\widetilde{A}$, y and λ.

Ensure: The solution vector $\widetilde{x}$ for the problem (4)

Initialize: $\widetilde{x} = d = \mathbf{0} \in \mathbb{R}^{\widetilde{N}}$ and $R = -y$.

while not converged **do**

In parallel on P processors

Randomly pick i from the index set $\{1, 2, ..., \widetilde{N}\}$.

Compute the ith gradient: $g(\widetilde{x})_i = \widetilde{A}_i^T R$.

Get $\Delta\widetilde{x}_i$: $\Delta\widetilde{x}_i = \Gamma_{\lambda/L}(\widetilde{x}_i - \frac{g(\widetilde{x})_i}{L}) - \widetilde{x}_i$.

$d_i = \lambda(|\widetilde{x}_i| - |\widetilde{x}_i + \Delta\widetilde{x}_i|) - g(\widetilde{x})_i \Delta x_i - \frac{1}{2}\|\widetilde{A}_i\|_2^2 \Delta x_i^2$.

for $t = (i - \omega/2)$ **to** $(i + \omega/2)$ **do**

if $d_i < d_t$ **then**

Return and perform the next iteration.

end if

end for

Update $\widetilde{x}_i$: $\widetilde{x}_i = \widetilde{x}_i + \Delta\widetilde{x}_i$.

Update R: $R = R + \Delta\widetilde{x}_i \widetilde{A}_i^T$.

Update d_i by the same way from 5th to 7th line.

end while

$$L = \max_{i=1,2,...,\widetilde{N}} [\nabla^2 F(\widetilde{x})]_{ii}. \tag{2.15}$$

For the Lasso problem, L can be calculated by:

$$L = \max_{i=1,2,...,\widetilde{N}} \|A_i\|_2^2. \tag{2.16}$$

Grouped Selection Strategy

The strategy of selecting potential coordinates in AGCD is to evaluate the descent of objective function for the selected candidate. If the selected coordinate i wins the competition in a group of candidates, x_i will be updated by $\Delta\widetilde{x}_i$. Otherwise this selection fails and the update in this iteration is terminated. In a multithreading environment, all the processors perform this process in parallel.

Then I need to estimate the descent of objective function for a specific coordinate. The descent of the objective function is stored and updated in d. For the ith coordinate, suppose that I have already obtained $\Delta\widetilde{x}_i$ and $g(\widetilde{x})_i$, d_i can be estimated by the following equation:

$$d_i = F(\widetilde{x}) - F(\widetilde{x} + \Delta\widetilde{x}_i e_i). \tag{2.17}$$

where e_i is a unit vector in the ith coordinate. When considering the Lasso problem, d_i can be rewritten as the following formula:

$$\lambda(|\widetilde{x}_i| - |\widetilde{x}_i + \Delta\widetilde{x}_i|) + \frac{1}{2}(\|\widetilde{A}\widetilde{x} - y\|_2^2 - \|\widetilde{A}\widetilde{x} + \widetilde{A}_i^T\Delta\widetilde{x}_i - y\|_2^2). \tag{2.18}$$

Finally, d_i can be calculated by:

$$d_i = \lambda(|\widetilde{x}_i| - |\widetilde{x}_i + \Delta\widetilde{x}_i|) - \frac{1}{2}{\Delta\widetilde{x}_i}^2\|\widetilde{A}_i\|_2^2 - \Delta\widetilde{x}_i\widetilde{A}_i^T R, \tag{2.19}$$

where $\widetilde{A}_i^T R$ equals to $g(\widetilde{x})_i$ that has already been calculated previously. The value of $\|\widetilde{A}_i\|_2^2$ is also obtained when calculating the Lipschitz constant L in equation (2.16). Thus, the time complexity to update d_i is $O(1)$.

After d_i is updated, there is a competition between the ith coordinate and a group of ω candidates. If d_i wins, $\widetilde{x}_i$ will be updated. Otherwise the update of the ith coordinate in this iteration is terminated. Bradley et al.Kyrola *et al.* (2011) showed that the number of updated coordinates is at most $\widetilde{N}/2\rho$ in one iteration where ρ is the spectral radius of $\widetilde{A}^T\widetilde{A}$. AGCD divides the feature space into $\widetilde{N}/\omega$ groups, which

means there are at most $\widetilde{N}/\omega$ coordinates to be updated in each iteration. Therefore, I set the size of group ω to be 2ρ in AGCD. If ω is larger than $\widetilde{N}$, ω is set to be $\widetilde{N}$. The way I set the candidate group is to select ω number of coordinates that are close to the ith coordinate. Specifically, the index in the group starts from $i - \omega/2$ and ends at $i + \omega/2$. If $i - \omega/2 \leq 0$, it starts from 1 to ω. When $i + \omega/2 \geq \widetilde{N}$, it is from $i - \omega$ to i. If the ith coordinate wins the competition, the update is performed by the following three steps:

Step1: Update $\widetilde{x}_i : \widetilde{x}_i + \Delta\widetilde{x}_i$.

Step2: Update $R : R = R + \Delta\widetilde{x}_i\widetilde{A}_i^T$.

Step3: Update d_i by equation (2.13), (2.14) and (2.19).

Fig 2.1 illustrates the process of group selection and asynchronous update flowchart with two threads. Although d_i has already been updated at the beginning, AGCD still needs to perform Step 3 in the above updating rules because I intend to minimize the effects of un-updated winners' d_i to the competition of other candidates. I still take the example in Figure 1 to illustrate this. After the 2nd coordinate wins the competition, AGCD performs Step 1 and Step 2 to update $\widetilde{x}_i$ and R. However, d_i is not the current descent of object function since x_i and R have changed. In the next iteration, if AGCD selects the 1st coordinate to evaluate, x_1 might still not be updated since $d_2 > d_1$ in the last iteration. Although d_1 is updated in the next iteration, x_1 still has a lower chance to be updated since x_2 is the winner last time and d_2 is the "winning distance". Because of asynchronous characteristic of AGCD, it is not guaranteed that all the d_i are updated to the newest one. AGCD makes all the winners' d_i updated to newest value to minimize the effects of winners to competitions of other candidates.

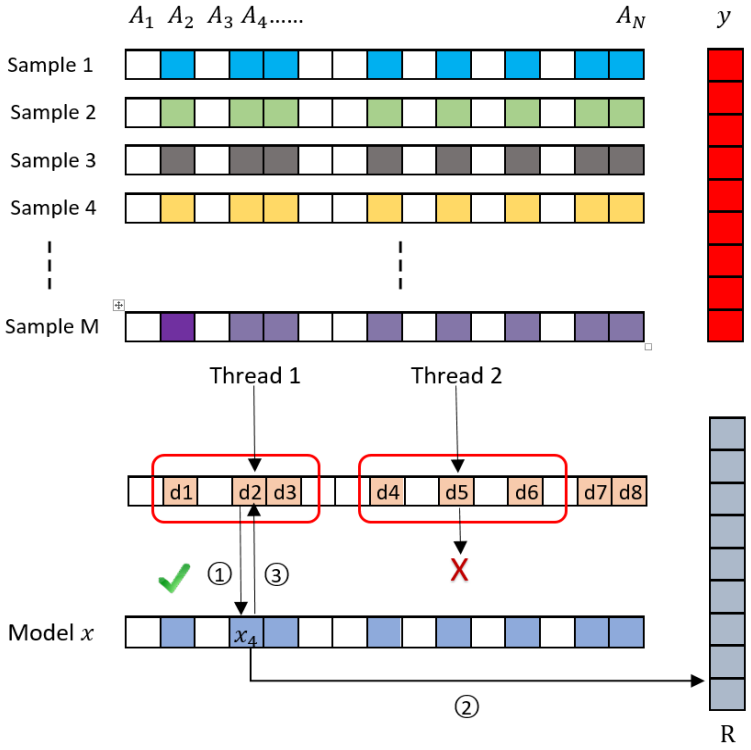


Figure 2.1: Illustration of AGCD with Two Threads. The White Blocks in Each Sample Represent Inactive Features Discarded by Screening Rules. Thread 1 Chooses the 2nd Active Feature to Evaluate. Firstly, d_2 Is Updated and Evaluated in a Group of $\{d_1, d_2, d_3\}$. d_2 Wins the Competition and I Follow the Three Steps to Update x_4, R and d_2. Thread 2 Selects the 5th Candidate. However, It Fails in the Competition and Update is Terminated for Thread 2 in this Iteration.

Discussion

I apply atomic operations to avoid the synchronization among threads when updating x_i, R and d_i. Since x, R and d_i are global variables, it is necessary to add locks on these shared variables when multiple threads attempt to update them simultaneously. However, updating a single variable and locking all the variables is not an efficient strategy since all the other threads have to wait for one thread to finish its job. I employ atomic operations to write the global variables atomically without any locks. Liu *et al.* (2014) and Hsieh *et al.* (2015) have observed empirical convergence when applying "atomic writes" on updating the shared variables.

AGCD adopts a grouped selection strategy to update the solution vector by choosing the candidate that minimizes the objective function with the most descent. In the random selection strategy used in parallel SCD solvers, a number of processors update the solution vector asynchronously, which is more likely to result in the divergence of the optimization problem in a small feature space. In AGCD, the update of solution variables is not as frequent as in the random selection strategy. The selected coordinate has to beat a group of candidates to get the chance to update. In fact, the selected coordinates will not be updated in most of the iterations since it failed in the competition. However, this does not mean that the computation spent in a failed candidate is a waste of time. Although x_i is not updated, it updates the descent value d_i for that coordinate. Suppose x_i is updated, it means that R is changed. As a result, all the elements in d should be updated by (14). Therefore, updating d concurrently is critical to guarantee the accurate result of competition in the grouped selection strategy.

2.6 Convergence Analysis

In this section, I analyze the convergence of the proposed parallel framework. PSAFE and PDPP are safe screening rules and it is guaranteed that all the discarded features have zero coefficients in the solution. PSR is a heuristic screening method but I can ensure the correctness of result by checking the KKT condition. AGCD can be safely applied on the reduced feature matrix $\widetilde{A}$ to optimize the problem (4). Therefore, the proposed parallel framework will work if I prove the convergence of AGCD.

I follow the same way in Shalev-Shwartz and Tewari (2011) to rewrite the objective function (2.1) into an equivalent problem with a twice-differentiable regularizer:

$$\min_{\hat{x}} \frac{1}{2}\sum_{j=1}^{M}(\hat{a}_j^T\hat{x} - y_j)^2 + \lambda\sum_{i=1}^{2N}\hat{x}_i : \hat{x} \in \mathbb{R}^{2N}, \tag{2.20}$$

where a_j denotes the jth row of A and the feature space is duplicated as: $\hat{a}_j = [a_j; -a_j]$ and $\hat{a}_j \in \mathbb{R}^{2N}$. Once the optimal solution $\hat{x}^*$ of equation (2.20) is obtained, I can recover the solution vector x^* of (1) by $x_i^* = \hat{x}_{d+i}^* - \hat{x}_i^*$. I denote the objective function $F(x)$ equal to (15) in the convergence analysis part.

Definition

Let $F(x) : \mathbb{R}^{2N} \to \mathbb{R}$ be a convex function. Assume that there exists $\beta > 0$, for all x and Δx updated in parallel, I have the following rule:

$$F(x + \Delta x) \leq F(x) + \Delta x^T \nabla F(x) + \frac{\beta}{2}\Delta x^T A^T A \Delta x.$$

I denote $\beta = 1$ for the square loss function and $\beta = \frac{1}{4}$ for the logistic loss function. Let $\hat{d} = [\hat{d}_1, \hat{d}_2, ...\hat{d}_{2N}]$ represent the potential candidates updated by (12). Δx denotes the collective update of x in one iteration. Δx_i is equal to zero when $\hat{d}_i$ fails the competition where $i \in (1, 2N)$.

When there is only one coordinate updated at the same time, I have $\Delta x = (\Delta x_i)e_i$ and e_i is a unit vector in the ith coordinate. The process of optimization is the same as the sequential coordinate descent when one coordinate is updated at each iteration. It was shown in Shalev-Shwartz and Tewari (2011) that the sequential coordinate descent converges by the following bound:

$$E[F(x^{(K)})] - F(x^*) \leq \frac{N(\beta\|x^*\|_2^2 + 2F(x^{(0)}))}{K+1}, \tag{2.21}$$

where $F(x^{(K)})$ is the output after K iterations. The convergence analysis of AGCD is the same as sequential coordinate descent if only one coordinate is updated in each iteration.

When there are more than one candidates winning the competition at the same time, I summarize the main analysis in the following theorem:

Theorem

Let x^* be the solution of F(x) and $x^{(K)}$ be the output of AGCD after K iterations. Let P be the number of processors and Φ denotes the maximum number of candidates to be updated in one iteration. Suppose $F(x)$ satisfies the assumption of Definition 1; let $\epsilon = \frac{(\Phi-1)(\rho-1)}{2N-1} < 1$ and ρ be the spectral radius of A^TA. I have

$$E[F(x^{(K)})] - F(x^*) \leq \frac{N(\beta\|x^*\|_2^2 + \frac{2}{1-\epsilon}F(x^{(0)}))}{(K+1)\Phi}.$$

Proof

I use a similar technique in Kyrola *et al.* (2011) to prove this theorem. Let Θ denote the index set that collects the winners of competitions in one iteration and

$\Phi = |\Theta|$. Based on the assumption of Definition 1, I have

$$
\begin{aligned}
& E_\Theta[F(x+\Delta x) - F(x)] \\
& \leq E_\Theta[\Delta x^T \nabla F(x) + \frac{\beta}{2}\Delta x^T A^T A \Delta x] \\
& = \Phi E_i[\Delta x_i \nabla F(x)_i + \frac{\beta}{2}(\Delta x_i)^2] \\
& + \frac{\beta}{2}\Phi(\Phi-1)E_{i,j:i\neq j}[\Delta x_i (A^T A)_{i,j} \Delta x_j] \\
& = \Phi E_i[\Delta x_i \nabla F(x)_i + \frac{\beta}{2}(1 + \frac{(\Phi-1)(\rho-1)}{2N-1})(\Delta x_i)^2] \\
& = \Phi E_i[\Delta x_i \nabla F(x)_i + \frac{\beta}{2}(1+\epsilon)(\Delta x_i)^2].
\end{aligned}
$$

Let $\rho = \max_{\mu:\mu^T\mu=1} \mu^T(A^TA)\mu$. $E_{i,j:i\neq j}[\Delta x_i (A^TA)_{i,j}\Delta x_j]$ is bounded by ρ in terms of $E_i[\Delta(x_i)^2]$ where i is chosen uniformly at random from $\{1, ..., 2N\}$. The rest of proof follows the same way in Shotgun Kyrola *et al.* (2011)'s convergence analysis to obtain the result of Theorem 1. I omit it for brevity.

In Shotgun, it was shown that the number of processors should satisfy $P \leq P^* \approx \frac{N}{2\rho}$ and the experiment demonstrates that Shotgun diverges as P exceeds P^*. As I discussed in section 3.5, the size of each group is set to be $\omega = 2\rho$. The maximum number of updated coordinates in one iteration is $\Phi = \frac{N}{\omega}$ which satisfy the above constraint. In the real cases, the number of updated candidates is much smaller than P since most of the updates happen in the calculation of d. In the grouped selection strategy of AGCD, each coordinate has a low probability to be updated among ω candidates. As a result, P can be equal to or larger than $\frac{N}{2\rho}$ in the real application of AGCD. I demonstrate it in the experiment that AGCD encountered the cases that the number of processors is larger than the number of active features while AGCD still converged to the optimal value.

Table 2.1: A Comparison of PDPP+AGCD and EDPP+SLEP along a Sequence of 100 Parameter Values on 0.5 Million ADNI Dataset

EDPP+SLEP			PDPP+AGCD		
λ/λmax	nnz in the solution	Objective function	λ/λmax	nnz in the solution	Objective function
1.0	0	373.0	1.0	0	373.0
0.95	4	372.9657	0.95	4	372.9657
0.9	8	372.6708	0.9	8	372.6708
0.8	26	370.5095	0.8	26	370.5095
0.75	46	368.3644	0.75	46	368.3644
0.7	69	364.2990	0.7	69	364.2990
0.6	134	351.0384	0.6	134	351.0384
0.55	169	341.9249	0.55	169	341.9249
0.5	216	327.9936	0.5	216	327.9936
0.4	307	292.8738	0.4	307	292.8738
0.35	364	272.4728	0.35	364	272.4728
0.3	419	244.0223	0.3	418	244.0223
0.2	508	179.7974	0.2	508	179.7974
0.15	571	145.1732	0.15	569	145.1732
0.1	650	98.9118	0.1	651	98.9118

2.7 Experimental Results

In this section, I conduct several experiments to evaluate the convergence and speedup of the proposed framework on the following four data sets: ADNI [2] , MNIST LeCun *et al.* (1998), rcv1 Lewis *et al.* (2004) and news20 Keerthi *et al.* (2005). The Alzheimer's Disease NeuroimagingInitiative (ADNI) is a real biomedical dataset collected from neuroimaging and genomic data from elderly individuals across North America, including 809 patients of Alzheimer's disease with 5,906,152 features, involving a 80 GB feature matrix with 42 billion nonzeros. For MNIST, rcv1 and news20, I use the training dataset obtained from LIBSVM data set repository [3] to construct the feature data matrices and response vectors. I compare our method with the state-of-the-art algorithms like PBCD Richtárik and Takáč (2016) and ASYSCD Liu *et al.* (2014). All the experiments are carried out on an Intel (R) Xeon (R) 48-core machine with 2.50 GHZ processors and 256 GB of globally addressable memory. I employ OpenMP as the parallel framework and all the methods are implemented in C++ for fair comparisons.

Accuracy Evaluation

In this experiment, I examine the accuracy of solution vectors in the proposed method. I perform PDPP+AGCD along a sequence of 100 parameter values equally spaced on the linear scale of $\lambda/\lambda_{\max}$ from 0.1 to 1. To make a comparison, I perform EDPP using SLEP Liu *et al.* (2009) as the solver on the same sequence. In SLEP, I force the "LeastR" function to run 500 iterations. AGCD also executes 500 iterations using 48 threads. Experiments are conducted on ADNI data sets. I choose the volume of the right pallidum as the response, including 747 samples by removing samples

[2] http://www.adni-info.org

[3] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

Table 2.2: Data Statistics

Dataset	Number of features	Number of samples
MNIST	780	60000
news20	62061	15935
rcv1	47236	15564
ADNI_1m	1000000	717
ADNI_2m	2000000	717
ADNI_3m	3000000	717
ADNI_5.9m	5906152	717

without labels. The volumes of brain regions are extracted from each subject's T1 MRI scan using Freesurfer [4] . I randomly select 0.5 million features from ADNI to construct the feature matrix and normalize the matrix using the "zscore" function in Matlab . The experimental result is shown in Table 2.1.

I report the result of 15 parameter values from 100 parameters. The first column in both methods is the position of the parameter in the sequence. The third column shows the remaining number of features $\widetilde{N}$ after applying screening rules. Table 2.1 shows that the optimal value obtained by the PDPP+AGCD and the number of nonzero in the solution is the same as that of EDPP+SLEP. When $\lambda/\lambda_{\max}$ is higher than 0.8, the remaining features after screening is less than the number of threads. However, PDPP+AGCD is still able to converge to the optimal value.

[4]http://freesurfer.net/

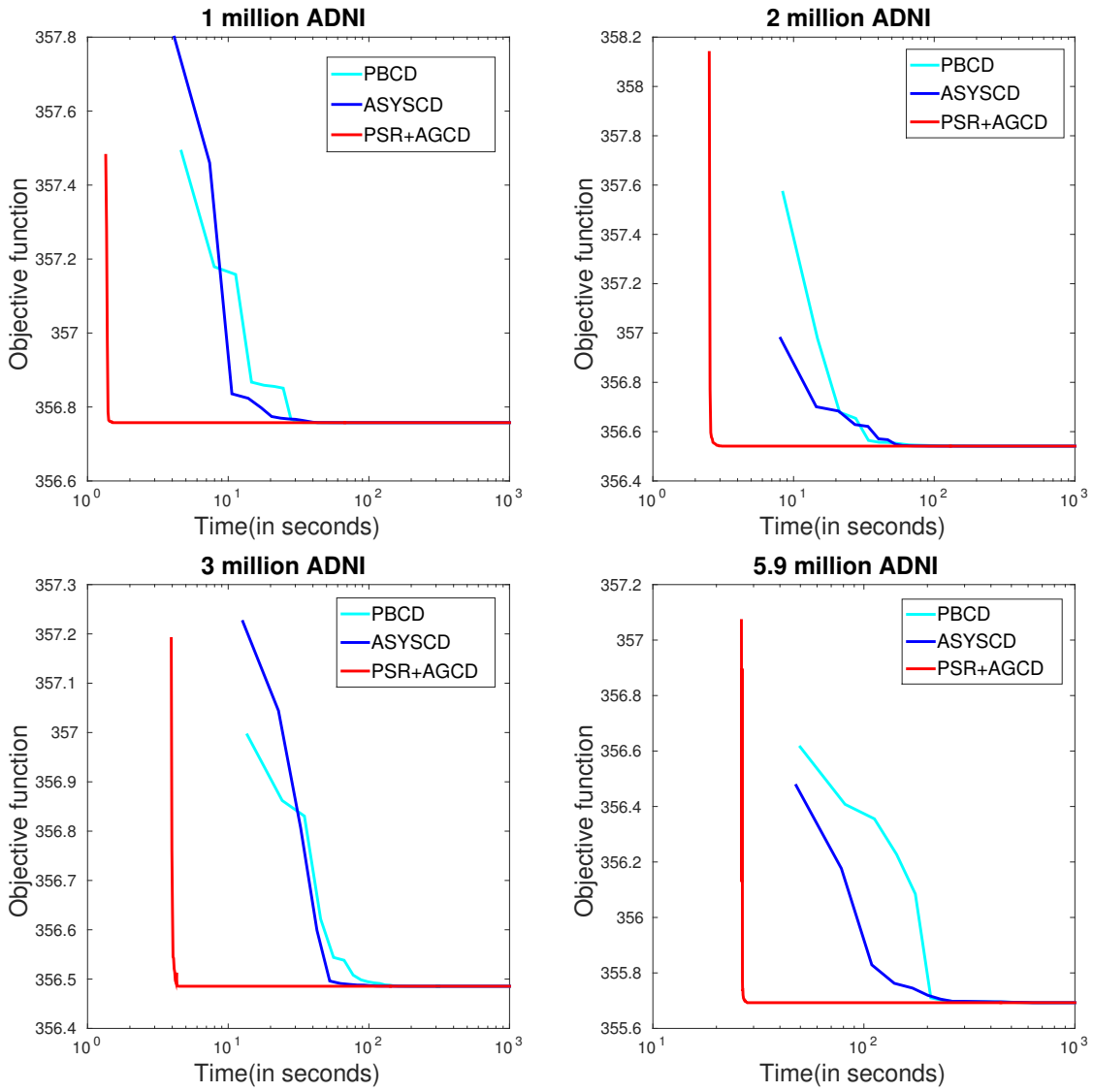


Figure 2.2: Convergence Comparison of Different Solvers When $\lambda = 0.8\lambda_{\max}$

Convergence Comparison

In this experiment, I evaluate the convergence property of the proposed parallel methods. I conduct the experiment on PSR+AGCD and compare with state-of-the-art parallel solvers: PBCD and ASYSCD. I choose two different λ value: $0.8\lambda_{\max}$ and $0.6\lambda_{\max}$ to estimate the convergence of above methods. To prevent PSR from

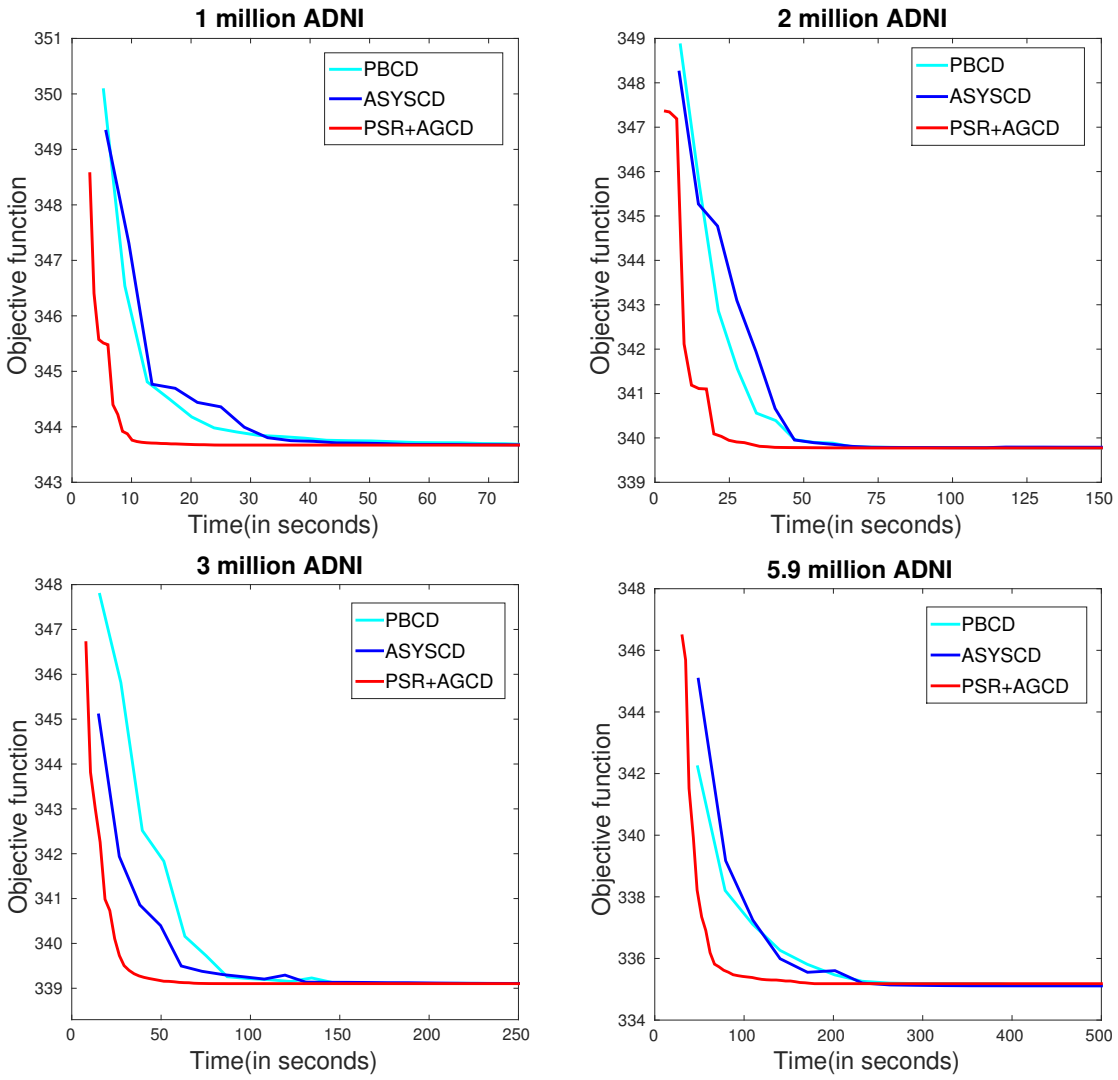


Figure 2.3: Convergence Comparison of Different Solvers When $\lambda = 0.6\lambda_{\max}$

discarding active features, I check the KKT condition to ensure the correctness of screening results. To estimate the scalability of above algorithms, the number of cores is varied from 1 to 32: 1, 2, 4, 8, 16 and 32. For different number of cores, I show the time of optimization that solvers converged to the same optimal values with 48 cores. I evaluate the efficiency of parallel solvers on four ADNI dataset: ADNI_1m, ADNI_2m, ADNI_3m and ADNI_5.9m where feature dimension is varied from 1 million to 5.9 million. I choose the volume of hippocampus in patients as

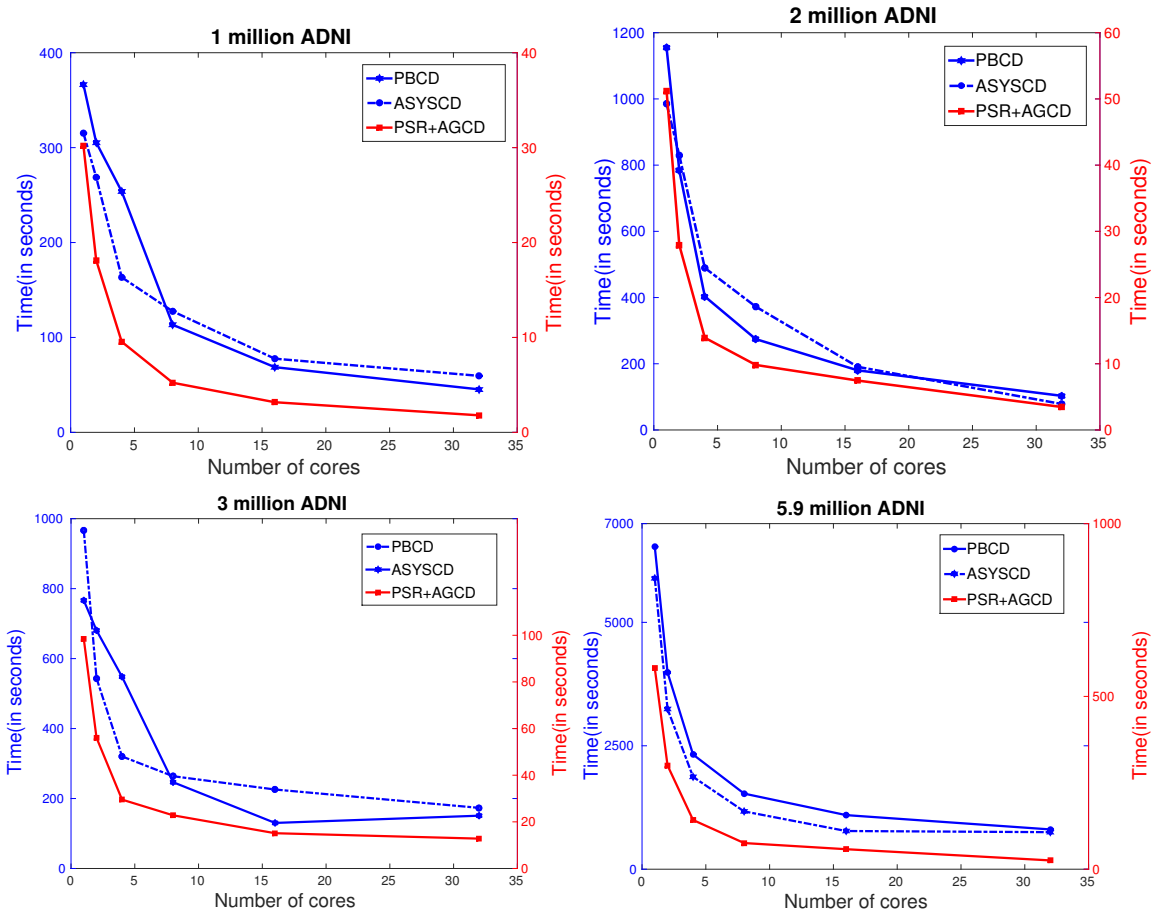


Figure 2.4: Speedup Comparison Different Solvers When $\lambda = 0.8\lambda_{\max}$

the response, including 717 samples from the original dataset. I show details of data sets in Table 2.2 and results of comparison in Fig 2.2, Fig 2.4, Fig 2.3 and Fig 2.5,respectively.

In the Fig 2.2 and Fig 2.3, I evaluate the convergence in terms of time using 48 cores. Note that I use log-scale in x-axis when $\lambda = 0.8\lambda_{\max}$. As observed from the figure, the objective function in PSR+AGCD converged faster than other solvers since most of inactive features are discarded. Most of time is spent in the screening part. When $\lambda = 0.6\lambda_{\max}$, only part of inactive features are discarded by screening but PSR+AGCD still has superior performances.

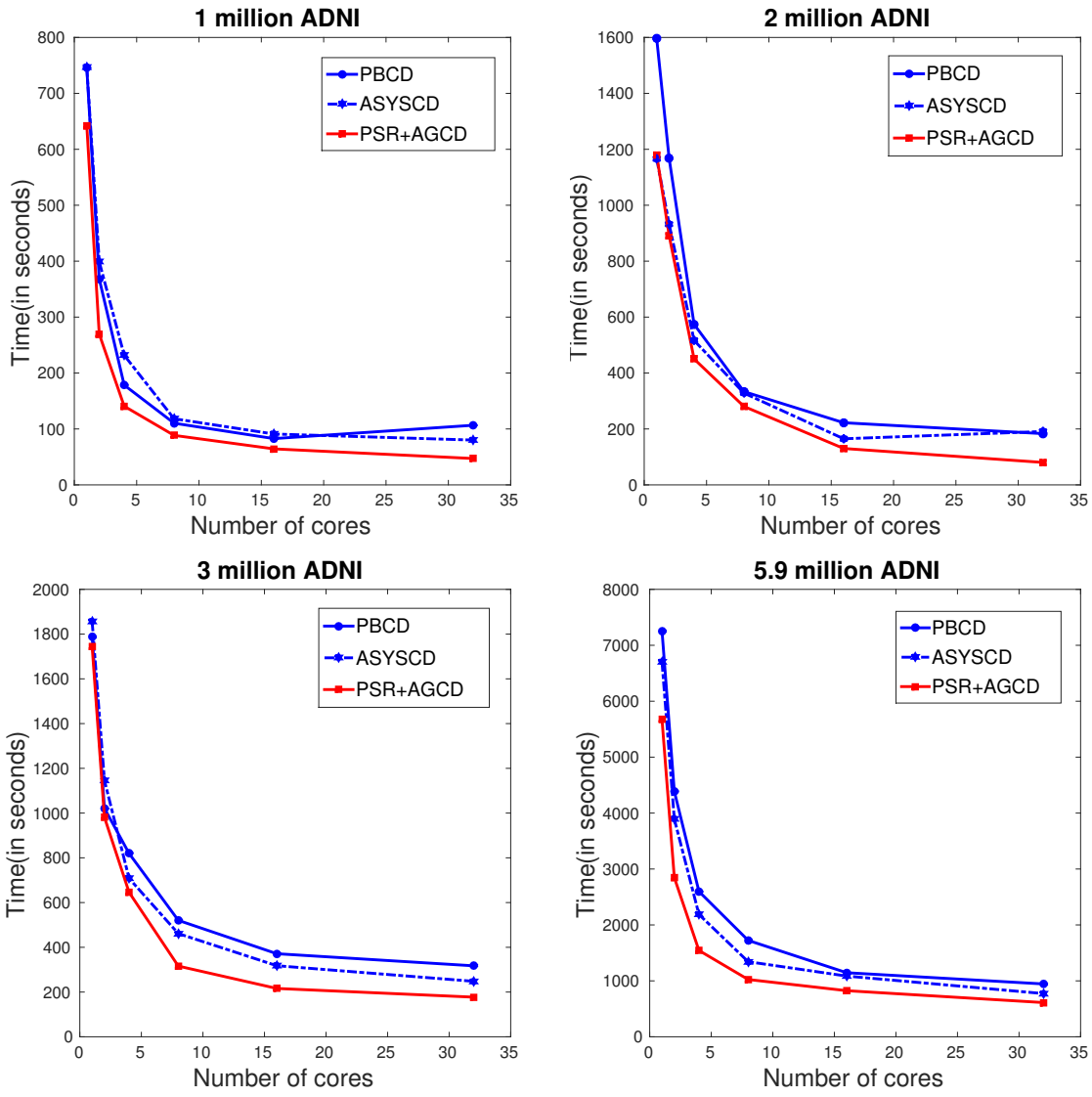


Figure 2.5: Speedup Comparison Different Solvers When $\lambda = 0.6\lambda_{\max}$

The Fig 2.4 and Fig 2.5 show the time of different solvers that converged to the same optimal value of 48 cores when varying the number of cores from 1 to 32. Note that I use two y-axis when λ equals to $0.8\lambda_{\max}$. PBCD and ASYSCD use the left y-axis while PSR+AGCD uses the right one. PSR+AGCD outperforms the other solvers when varying the number of cores and the running time of PSR+AGCD is reduced when there are more cores.

Table 2.3: Efficiency Comparison along a Sequence of Parameter Values on MNIST

Dataset: MNIST							
λ	Method	Time spent in different number of cores (in minutes)					
		1	2	4	8	16	32
100	ASYSCD	131.08	76.25	62.78	39.17	26.7	23.33
	PDPP+AGCD	6.37	3.61	2.43	1.49	1.02	0.87
	Speedup	**20.58**	**21.12**	**25.84**	**26.29**	**26.178**	**26.82**
200	ASYSCD	283.43	182.36	116.86	61.96	45.76	38.37
	PDPP+AGCD	10.9	6.25	3.96	1.96	1.48	1.17
	Speedup	**26.01**	**29.18**	**29.51**	**31.61**	**30.92**	**32.79**
400	ASYSCD	571.46	364.21	238.65	129.24	92.58	71.71
	PDPP+AGCD	17.14	11.07	7.13	3.86	2.53	1.76
	Speedup	**33.34**	**32.90**	**33.47**	**33.48**	**36.59**	**40.74**

Time Efficiency

The advantage of the proposed parallel framework is to solve the Lasso problem along a sequence of parameter values. In this experiment, I perform PDPP+AGCD along a sequence of parameter values equally spaced on the linear scale of λ/λmax from 0.1 to 1. I vary the length of parameter sequences as 100, 200 and 400. As a comparison, ASYSCD is performed on the same sequence. The experiment is conducted at three different data sets: MNIST, news20 and ADNI_2m. Detailed information about data sets is in Table 2.2. To evaluate the scalability of both methods, I vary the number of cores as: 1, 2, 4, 8, 16 and 32. The result of comparison is presented in Table 2.3, Table 2.4 and Table 2.5,respectively.

The experimental results in the tables show that more parameters lead to higher

Table 2.4: Efficiency Comparison along a Sequence of Parameter Values on ADNI_2m

Dataset: ADNI_2m							
λ	Method	Time spent in different number of cores (in minutes)					
		1	2	4	8	16	32
100	ASYSCD	3205.13	1892.34	1105.21	756.23	435.64	324.81
	PDPP+AGCD	48.34	23.63	12.21	8.28	5.34	4.03
	Speedup	**66.30**	**80.08**	**90.52**	**91.33**	**81.58**	**80.59**
200	ASYSCD	5614.21	3217.91	1821.87	1345.21	826.51	692.87
	PDPP+AGCD	63.32	34.07	18.81	12.99	7.44	5.79
	Speedup	**88.66**	**94.44**	**96.86**	**103.56**	**111.09**	**119.67**
400	ASYSCD	11275.34	6328.35	3812.87	2315.34	1521.54	1296.54
	PDPP+AGCD	95.42	52.17	33.10	18.18	11.06	9.57
	Speedup	**118.16**	**119.57**	**115.19**	**127.35**	**137.57**	**135.47**

speedup for PDPP+AGCD compared to ASYSCD. PDPP+AGCD achieved 137 folds speedup in ADNI_2m dataset, 101 folds in news20, and 40 folds in MNIST over ASYSCD with 400 parameters. When using more cores, speedups of our method tend to increase. Thus, in terms of speedup, PDPP+AGCD favors more cores and sequences with more parameter values.

Scalability

To estimate the scalability of proposed parallel methods, I perform PSR+AGCD and PDPP+AGCD on 1, 2, 4, 8, 16, 32 and 48 cores to observe the speedup. I give the definition of speedup by the following criterion:

$$\text{speedup} = \frac{\text{time spent on } P \text{ processors}}{\text{time spent on a single processor}}.$$

Table 2.5: Efficiency Comparison along a Sequence of Parameter Values on News20

Dataset: news20							
λ	Method	Time spent in different number of cores (in minutes)					
		1	2	4	8	16	32
100	ASYSCD	2736.52	1491.26	762.57	403.06	265.71	194.27
	PDPP+AGCD	40.22	20.69	10.60	5.59	3.67	2.50
	Speedup	**68.04**	**72.07**	**71.93**	**72.03**	**72.40**	**77.67**
200	ASYSCD	5528.48	2813.78	1525.03	804.68	552.83	358.35
	PDPP+AGCD	64.40	30.07	16.33	8.47	5.69	3.63
	Speedup	**85.84**	**93.56**	**93.38**	**95.38**	**97.02**	**98.77**
400	ASYSCD	10437.35	5480.47	2812.21	1565.12	1057.30	662.792
	PDPP+AGCD	120.38	64.31	29.05	15.88	10.42	6.67
	Speedup	**86.70**	**85.21**	**96.79**	**98.54**	**101.51**	**99.47**

In this experiment, I employ both methods on 5.9 million ADNI and rcv1 data sets, respectively. PDPP+AGCD is carried out along a 100 linear-scale sequence of parameter values from 0.1 to 1. For PSR+AGCD, I set the parameter λ to be $0.8\lambda_{\max}$ in the optimization. Fig 2.6 presents the result. PDPP+AGCD is more scalable than PSR+AGCD and achieves approximate 17 and 11.5 folds speedup with 48 cores in ADNI_5.9m and rcv1 data sets, respectively.

2.8 Summary

This study proposed a parallel framework to solve the ℓ_1-regularized minimization problem on huge dimensional datasets. I introduce screening rules into a parallel platform to discard the inactive features before optimization, accelerating the whole

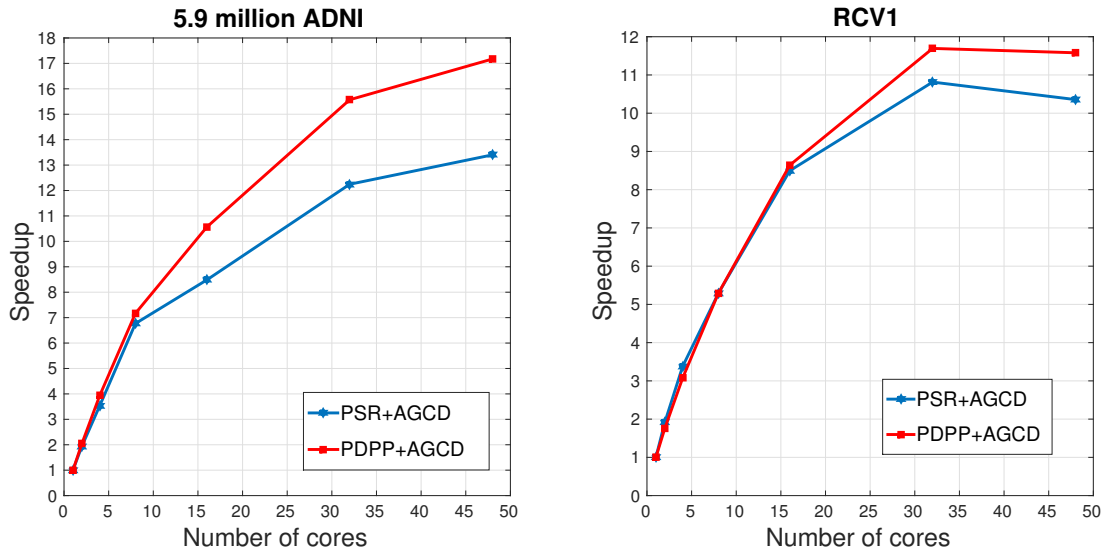


Figure 2.6: Speedup of Proposed Methods on 5.9 Million ADNI and Rcv1 Data Sets, Respectively

learning process significantly. Then the problem boils down to solve the optimization problem on a multithreading environment with a small number of feature space. A grouped selection strategy is proposed to select the candidates that minimize the objective function with the largest descent. Experiments demonstrate the efficiency and effective of proposed methods by conducting on different data set of real word applications. I present how to scale up the sparse regression model in a multithreading environment. In the next chapter, I will extend it to a distributed environment where the optimization is conducted with multiple computation nodes.

Chapter 3

OPTIMIZE THE DISTRIBUTED SPARSE REGRESSION MODELS AND STUDIES OF RISK GENETIC FACTOR FOR ALZHEIMER'S DISEASE

3.1 Introduction

Starting from this chapter, I present some real world applications where conducting the sparse learning process in a distributed environment. In this chapter, I propose a distributed framework for large-scale collaborative Imaging genetics studies across multiple research institutions.

Alzheimer's disease (AD) is a severe and growing worldwide health problem, and it doubles in frequency every five years after age sixty. Patients are progressively impaired in memory and other aspects of cognitive performance Burns (2009) and may become incapable of self-care as the disease advances. By 2050, the US may have 13.5 million clinical AD patients, with cost of care of about 1.1 trillion Association *et al.* (2011). Many techniques have been developed to investigate AD, such as magnetic resonance imaging (MRI), genome-wide association studies (GWAS) and whole genome sequencing (WGS). These techniques can help to identify preclinical and clinical AD patients relative to cognitively normal elderly controls Dickerson *et al.* (2001). Computer-aided diagnosis techniques are promising to help preclinical AD research, especially given the vast number of data available in 3D brain images. When a big amount of features are measured from a small number of subjects, it is often necessary to reduce their dimensions. Therefore, sparse learning Liu *et al.* (2009) is a powerful tool to represent local features effectively and concisely, helping image content analysis.

GWAS are achieving great success in finding single nucleotide polymorphisms (SNPs) associated with AD Harold *et al.* (2009). For example, APOE is a highly prevalent AD risk gene, and each copy of the adverse variant is associated with a 3-fold increase in AD risk Corder *et al.* (1993). The Alzheimer's Disease Neuroimaging Initiative (ADNI) collects neuroimaging and genomic data from elderly individuals across North America, including people with Alzheimer's disease, and these data are analyzed by geographically distributed investigators Toga *et al.* (2010). The ENIGMA Thompson *et al.* (2014) Consortium works as a large-scale collaborative network consisting of 185 research institutions around the world, analyzing neuroimaging and genomic data from over 33,000 subjects, from 35 countries. However, processing and integrating genetic data across different institutions is challenging. Each participating institution may wish to collaborate with others, but often legal or ethical regulations restrict access to individual data, to avoid compromising data privacy. A common goal is to identify common genetic variants associated with brain measures or with disease, based on data from different institutions, while preserving privacy.

Some studies, such as ADNI, share genomic data publicly under certain conditions, but more commonly, each participating institution may be required to keep their genomic data private, so collecting all data together may not be feasible. To deal with this challenge, I proposed a novel distributed framework, termed Local Query Model (LQM), to perform the Lasso regression analysis—a popular sparse learning technique — in a distributed manner. With LQM, collaborating institutions can learn and identify genetic risk factors without accessing others' data. However, applying LQM for model selection—such as stability selection—can be very time consuming on a large-scale data set. To speed up the learning process, I proposed a family of distributed safe screening rules (D-SAFE and D-EDPP) to identify irrelevant features and remove them from the optimization without sacrificing accuracy. Next, LQM is

employed on the reduced data matrix to train the model so that each institution obtains top risk genes for AD by stability selection on the learnt model without revealing its own data set. I evaluate our method on the ADNI GWAS data, which contains 809 subjects with 5,906,152 SNP features, involving a 80 GB data matrix with approximate 42 billion nonzero elements, distributed across three research institutions. Empirical evaluations demonstrate a speedup of 66-fold gained by D-EDPP, compared to LQM without D-EDPP. Stability selection results show that proposed framework ranked *APOE* as the first risk SNPs among all features, offering a powerful feature selection tool to study AD and its early symptom.

3.2 Data Processing

ADNI GWAS Data

The ADNI GWAS data contains genotype information for each of the 809 ADNI participants, which consist of 128 patients with AD, 415 with mild cognitive impairment (MCI), 266 cognitively normal (CN). To store statistically relevant SNPs called using Illuminas CASAVA SNP Caller, the ADNI WGS SNP data is stored in variant call format (VCF) for storing gene sequence variations. SNPs at approximately 5.9 million specific loci are recorded for each participant. I encode SNPs with the coding scheme in Sasieni (1997) and apply Minor Allele Frequency (MAF) < 0.05 and Genotype Quality (GQ) < 45 as two quality control criteria to filter high quality SNPs features. I follow the same coding scheme and quality control of Yang *et al.* (2015).

Data Partition

Lasso Tibshirani (1996) is a widely-used regression technique to find sparse representations of data, or predictive models or predictive models based on an efficient

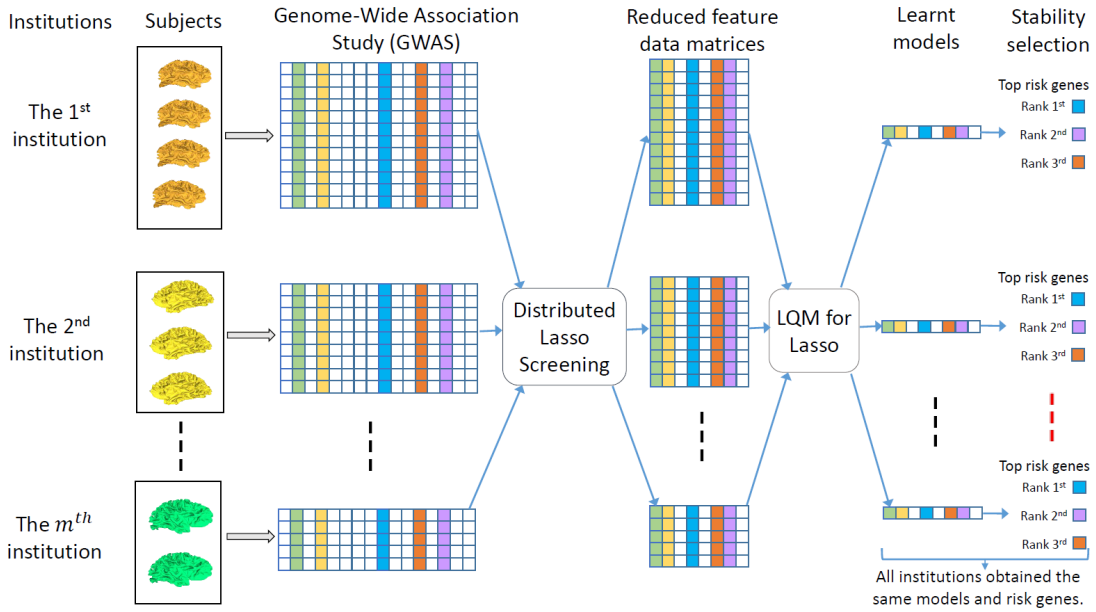


Figure 3.1: The Streamline of Proposed Distributed Framework.

or low-dimensional predictor set. Standard Lasso takes the form of

$$\min_x \frac{1}{2}||Ax - y||_2^2 + \lambda||x||_1 : x \in \mathbb{R}^p, \tag{3.1}$$

In this chapter, I use A denotes genomic data sets distributed across different institutions, y is the response vector (e.g., hippocampus volume or disease status), x is sparse representation—shared across all institutions—that I need to learn, and λ is a positive regularization parameter.

Suppose that I have m participating institutions. For the ith institution, I denote its data set by (A_i, y_i), where $A_i \in \mathbb{R}^{n_i \times p}$, n_i is the number of subjects in this institution, p is the number of features, and $y_i \in \mathbb{R}^{n_i}$ is the corresponding response vector, and $n = \sum_i^m n_i$. I assume p is the same across all m institutions. Our goal is to apply Lasso regression to rank the top risk genetic factors of the AD disease based on the distributed data sets (A_i, y_i), $i = 1, 2, ..., m$.

3.3 Methods

Fig 3.1 illustrates the general idea of our distributed framework. Suppose that each institution maintains the ADNI genome-wide data for a few subjects. (In reality, all the ADNI data is now available for download from one site, but well-powered genome-wide association studies, such as ENIGMA Thompson *et al.* (2014), often pool 50 or more such genomic datasets from different sites; here I partition the ADNI data as an example). I first apply the distributed Lasso screening rule to pre-identify inactive features and remove them from the training phase. Next, I employ the Local Query Model on the reduced data matrices to perform collaborative analyses across different institutions without compromising data privacy. Finally, each institution obtains the learnt model and performs stability selection to rank the SNPs that may collectively affect AD. The process of stability selection is to count the frequency of nonzero entries in the solution vectors and select the most frequent ones as the top risk genes for AD. The whole learning procedure results in the same model for all institutions, and preserves data privacy at each of them.

Local Query Model

I apply a proximal gradient descent algorithm—the Iterative Shrinkage/Thresholding Algorithm (ISTA) Daubechies *et al.* (2004)—to solve the problem (3.1). I define the notation $g(x; A, y) = ||Ax - y||_2^2$ as the least square loss function. The general updating rule of ISTA is:

$$x^{k+1} = \Gamma_{\lambda t_k}(x^k - t_k \nabla g(x^k; A, y)), \tag{3.2}$$

where k denotes the iteration number, t_k denotes an appropriate step size in the kth iteration, and Γ is the soft thresholding operator Shalev-Shwartz and Tewari (2011)

defined by the following function:

$$\Gamma_\alpha(x) = sign(x) \cdot (|x| - \alpha)_+. \tag{3.3}$$

In view of (3.2), to solve (3.1), I need to compute the gradient of the loss function ∇g, which equals to $A^T(Ax - y)$. However, because the data set (A, y) is distributed to different institutions, I can not compute the gradient directly. To address this challenge, I propose a Local Query Model to learn the model x across multiple institutions without compromising data privacy.

In our study, each institution maintains its own data set (A_i, y_i) to preserve their privacy. To avoid collecting all data matrices $A_i, i = 1, 2, ..., m$ together, I can rewrite the problem (3.1) as the following equivalent formulation:

$$\min_x \sum_i^m g_i(x; A_i, y_i) + \lambda||x||_1 : i = 1, 2, ..., m, \tag{3.4}$$

where $g_i(x; A_i, y_i) = \frac{1}{2}||A_i x - y_i||_2^2$ is the least squares loss.

The key of LQM lies in the following decomposition:

$$\nabla g = A^T(Ax - y) = \sum_{i=1}^m A_i^T(A_i x - y_i) = \sum_{i=1}^m \nabla g_i. \tag{3.5}$$

I use "local institution" to denote all the participating institutions and "global center" to represent the place where intermediate results are calculated. For the ith local institution, the global center would query the gradient information with respect to a particular model x. In this case, the ith local center will compute $\nabla g_i = A_i^T(A_i x - y_i)$. Then, each local institution sends the partial gradient information of the loss function to the global center. After gathering all the gradient information, the global center can compute the accurate gradient with respect to x by adding all ∇g_i together and send the updated gradient ∇g back to all the local institutions to compute the model x. In the proposed framework, as LQM only acquires the partial

gradient from each local institution, it can properly maintain data privacy for all the institutions.

The master only servers as the computation center and does not store any data sets. Although the master gets g_i, it could not reconstruct A_i and y_i. Let g_i^k denote the kth iteration of g_i. Suppose x is initialized to be zero, $g_i^1 = -A_i^T y_i$ and $g_i^k = A_i(A_i^T x^k - y_i)$. I get $A_i^T A_i x$ by $g_i^k - g_i^1$ but A_i can not be reconstructed since updating and storing x only happens in the workers. As a result, LQM can properly maintain data privacy for all the institutions.

Distributed Safe Screening Rules for Lasso

As data are distributed to different institutions, I develop a family of distributed Lasso screening rule to identify and discard inactive features in a distributed environment. Suppose ith institution holds the data set (A_i, y_i), I summarize a distributed version of SAFE screening rules (D-SAFE) as follows:

Step 1: $Q_i = [A_i]^T y_i$, update$Q = \sum_i^m Q_i$ by LQM.

Step 2: $\lambda_{\max} = \max_j |[Q]_j|$.

Step 3: If $\left|[A]_j^T y\right| < \lambda - |||[A]_j||_2 ||y||_2 \frac{\lambda_{\max} - \lambda}{\lambda_{\max}}$, discard jth feature.

To compute $|||[A]_j||_2$ in Step 3, I first compute $H_i = |||[A_i]_j||_2^2$ and perform LQM to compute H by $H = \sum_i^m H_i$. Then, I have $|||[A_i]_j||_2 = \sqrt{H}$. Similarly, I can compute $||y||_2$ in Step 3. As the data communication only requires intermediate results, D-SAFE preserves the data privacy at each institution.

In many applications, the optimal value of λ is unknown. To tune the value of λ, commonly used methods such as cross validation need to solve the Lasso problem along a sequence of parameters $\lambda_0 > \lambda_1 > ... > \lambda_\kappa$, which can be very time-consuming. Enhanced Dual Polytope Projection (EDPP) Wang *et al.* (2013a) is a highly efficient

Algorithm 4 Distributed Enhanced Dual Polytope Projection (D-EDPP)

Require: A set of data pairs $\{(A_1, y_1), (A_2, y_2), ..., (A_n, y_n)\}$ and ith institution holds the data pair (A_i, y_i). A sequence of parameters: $\lambda_{\max} = \lambda_0 > \lambda_1 > ... > \lambda_\kappa$.

Ensure: The learnt models: $\{x^*(\lambda_0), x^*(\lambda_1), ..., x^*(\lambda_\kappa)\}$.

Perform the computation on n institutions. For the ith institution:

Let $R_i = A_i^T y_i$, compute $R = \sum_i^m R_i$ by LQM. Then I get $\lambda_{\max}$ by $||R||_\infty$.

$J = \arg\max_j |R|$, $v_i = [A_i]_J$ where $[A_i]_J$ is the Jth column of A_i.

Let $\lambda_0 \in (0, \lambda_{\max}]$ and $\lambda \in (0, \lambda_0]$.

$$\theta_i(\lambda) = \begin{cases} \frac{y_i}{\lambda_{\max}}, & \text{if } \lambda = \lambda_{\max}, \\ \frac{y_i - A_i x^*(\lambda)}{\lambda}, & \text{if } \lambda \in (0, \lambda_{\max}), \end{cases}$$

$T_i = v_i^T * y_i$, compute $T = \sum_i^m T_i$ by LQM.

$$v_1(\lambda_0)_i = \begin{cases} \text{sign}(T) * v_i, & \text{if } \lambda_0 = \lambda_{\max}, \\ \frac{y_i}{\lambda_0} - \theta_i(\lambda_0), & \text{if } \lambda_0 \in (0, \lambda_{\max}), \end{cases}$$

$v_2(\lambda, \lambda_0)_i = \frac{y_i}{\lambda} - \theta_i(\lambda_0)$, $S_i = ||v_1(\lambda_0)_i||_2^2$, compute $S = \sum_i^m S_i$ by LQM.

$v_2^\perp(\lambda, \lambda_0)_i = v_2(\lambda, \lambda_0)_i - \frac{<v_1(\lambda_0)_i, v_2(\lambda, \lambda_0)_i>}{S} v_1(\lambda_0)_i$.

Given a sequence of parameters: $\lambda_{\max} = \lambda_0 > \lambda_1 > ... > \lambda_\kappa$, for $k \in [1, \kappa]$, I make a prediction of screening on λ_k if $x^*(\lambda_{k-1})$ is known:

for j=1 **to** p **do**

$w_i = [A_i]_j^T(\theta_i(\lambda_{k-1}) + \frac{1}{2} v_2^\perp(\lambda_k, \lambda_{k-1})_i)$, compute $w = \sum_i^m w_i$ by LQM.

if $w < 1 - \frac{1}{2}||v_2^\perp(\lambda_k, \lambda_{k-1})||_2 ||[A]_j||_2$ **then**

I identify $[x^*(\lambda_k)]_j = 0$.

end for

safe screening rules to quickly identify irrelevant features along a sequence of parameters by utilizing the information of optimal solutions in the previous parameter, estimating the dual problem and geometric properties of Lasso regression problem and achieving about 200x speedups for real-world applications.

Since data are not shared among different institutions, each institution can only access its own data set. To address the problem of data privacy, I propose a distributed Lasso screening rule, termed Distributed Enhanced Dual Polytope Projection (D-EDPP), to identify and discard inactive features along a sequence of parameter values in a distributed manner. The idea of D-EDPP is similar to LQM. Specifically, to update the global variables, I apply LQM to query each local center for intermediate results–computed locally–and I aggregate them at global center. After obtaining the reduced matrix for each institution, I apply the proposed LQM to solve the Lasso problem on the reduced data set $\widetilde{A}_i$, $i = 1, ..., m$, which may contain a substantially reduced number of features. As explained in Chapter 3.3.1, solving Lasso on the reduced data sets by LQM will not compromise data privacy of each participating institution. I assume that j indicates the jth column in A, $j = 1, ..., p$, where p is the number of features. I summarize the proposed D-EDPP in Algorithm 4.

To speed this process up, I introduce a "warm-start" technique into our D-EDPP algorithm. First of all, I employ D-EDPP to identify the inactive features at λ_0. Then, I compute the optimal solution $x^*(\lambda_0)$ by solving Lasso on the reduced data matrix. In the next round, after obtaining a reduced matrix by screening, I use $x^*(\lambda_0)$ as the initial value rather than zero to solve the Lasso problem. The warm-start technique utilizes the model I solved in the last round to facilitate the optimization.

To calculate R, I apply LQM through aggregating all the R_i together in the global center by $R = \sum_i^m R_i$ and send R back to every institution. The same approach is used to calculate T, S and w in D-EDPP. The calculation of $||[A]_j||_2$ and

$||v_2^{\perp}(\lambda_k, \lambda_{k-1})||_2$ follows the same way in D-SAFE. Since only intermediate results are transferred among institutions, D-EDPP preserves the privacy.

The discarding result of λ_k relies on the previous optimal solution $x^*(\lambda_{k-1})$. Especially, λ_k equals to $\lambda_{\max}$ when k is zero. Thus, I identify all the elements of $x^*(\lambda_0)$ to be zero if k is equal to zero. When k is 1, I can perform screening rules based on $x^*(\lambda_0)$ to discard the inactive features.

Local Query Model for Lasso

To further accelerate the learning process, I apply FISTA Beck and Teboulle (2009); Peng *et al.* (2013) to solve the Lasso problem in a distributed manner. The convergence rate of FISTA is $O(1/k^2)$ compared to $O(1/k)$ of ISTA, where k denotes the iteration number. In this study, I integrate FISTA with LQM, termed as F-LQM, to solve the Lasso regression problem on the reduced matrix $\widetilde{A}_i$. I summarize the updating rule of F-LQM in kth iteration as follows:

$$\text{Step 1: } \nabla g_i^k = \widetilde{A}_i^T(\widetilde{A}_i x^k - y_i), \text{ update } \nabla g^k = \sum_i^m \nabla g_i^k \text{ by LQM.}$$

$$\text{Step 2: } z^k = \Gamma_{\lambda t_k}(x^k - t_k \nabla g^k) \text{ and } t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}.$$

$$\text{Step 3: } x^{k+1} = z^k + \frac{t_k - 1}{t_{k+1}}(z^k - z^{k-1}).$$

The matrix $\widetilde{A}_i$ denotes the reduced matrix for the ith institution obtained by the proposed D-EDPP rule. I repeat this procedure until a satisfactory global model is obtained. Step 1 calculates ∇g_i^k from local data $(\widetilde{A}_i, y_i)$. Then, each institution performs LQM to get the gradient ∇g^k based on (5). Step 2 updates the auxiliary variables z^k and step size t_k. Step 3 updates the model x. Similar to LQM, the data privacy of all the institutions are well preserved by F-LQM.

3.4 Experiment

I implement the proposed D-EDPP+LQM approach on a state-of-the-art distributed computing platform: Apache Spark [1] , which manages query processing with complex analytics on large clusters. To evaluate our proposed method, I report the efficiency through a study across three campuses: University of Southern California (USC), University of Michigan (UMich) and Arizona State University (ASU); clearly it could be interesting to scale this up to many multi-cohort datasets, as in ENIGMA Thompson *et al.* (2014). I show the efficiency of D-EDPP on a sequence of parameter values and employ stability selection with D-EDPP+LQM to determine top risk genetic factors associated with the disease AD.

Distributed Platform

Apache Spark is a fast and efficient distributed platform for large-scale data computing. Compared with Hadoop's two-stage disk-based MapReduce paradigm, Spark's multi-stage in memory design improves the efficiency up to 100 times for applications. Our Spark platform is deployed at three campuses: USC, UMich, and ASU. The off-campus project follows the master-workers pattern. Each institution maintains several machines as a worker node and one particular cluster in UMich works as the master node.

Comparison of Lasso With and Without D-EDPP Rule

I choose the volume of lateral ventricle as variables being predicted in trials containing 717 subjects by removing subjects without labels. The volumes of brain regions were extracted from each subject's T1 MRI scan using Freesurfer: http://freesurfer.net.

[1] http://spark.apache.org/

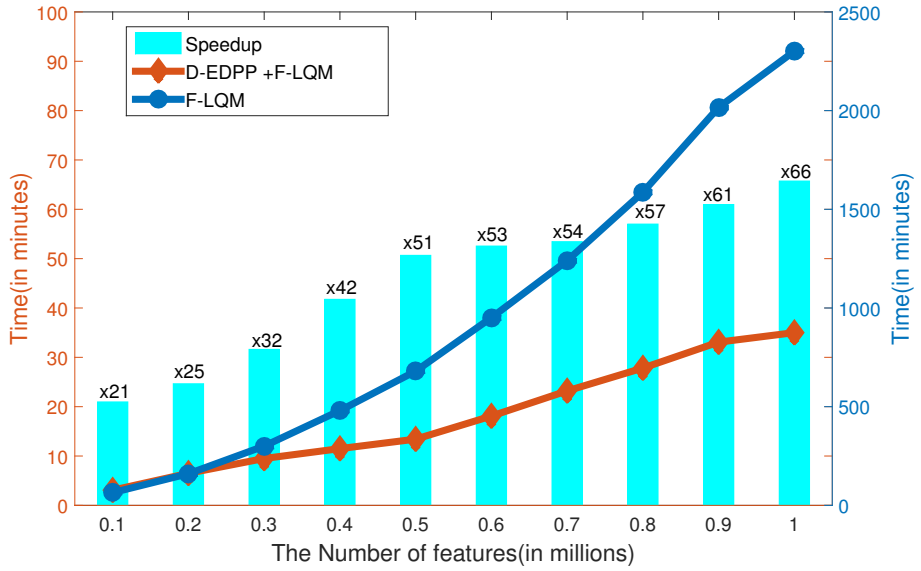


Figure 3.2: The Running Time Comparison of Lasso With and Without D-EDPP Rules.

I evaluate the efficiency of D-EDPP across three research institutions that maintain 326, 215, and 176 subjects, respectively. The subjects are stored as HDFS files. I solve the Lasso problem along a sequence of 100 parameter values equally spaced on the linear scale of $\lambda/\lambda_{\max}$ from 1.00 to 0.05. I randomly select 0.1 million to 1 million features by applying F-LQM since Beck and Teboulle (2009) proved that FISTA converges faster than ISTA. I report the performance of Lasso with and without D-EDPP rule in Fig 3.2. I achieved about a speedup of 66-fold compared to F-LQM.

Stability Selection For Top Risk Genetic Factors

I employ stability selection Meinshausen and Buhlmann (2010); Yang *et al.* (2015) with D-EDPP+F-LQM to select top risk SNPs from the entire GWAS data set with 5,906,152 features. I conduct four groups of trials using the diagnose at baseline, the volume of hippocampus, entorhinal cortex and lateral ventricle at baseline as the

Table 3.1: Top 5 Selected Risk SNPs Associated with Diagnose, the Volume of Hippocampal, Entorhinal cortex, and Lateral Ventricle at Baseline, Based on ADNI.

Diagnose at baseline				Hippocampus at baseline			
No.	Chr	RS_ID	Gene	No.	Chr	RS_ID	Gene
1	19	rs429358	APOE	1	19	rs429358	APOE
2	19	rs769449	APOE	2	8	rs34173062	SHARPIN
3	12	rs3136564	CD69	3	11	rs10831576	GALNT18
4	1	rs2227203	unknown	4	10	rs12412466	PPA1
5	20	rs6100558	PHACTR3	5	6	rs71573413	unknown
Entorhinal cortex at baseline				Lateral ventricle at baseline			
No.	Chr	RS_ID	Gene	No.	Chr	RS_ID	Gene
1	19	rs429358	APOE	1	Y	rs2261174	unknown
2	15	rs8025377	ABHD2	2	10	rs10994327	ANK3
3	Y	rs79584829	unknown	3	Y	rs62610496	unknown
4	14	rs41354245	MDGA2	4	1	rs2647521	AK5
5	3	rs55904134	unknown	5	1	rs2629810	SYT6

response variable for each group, respectively. In each trial, D-EDPP+F-LQM is carried out along a 100 linear-scale sequence of parameter values from 1 to 0.05. I simulate this 200 times and perform our method on 500 of subjects in each round. Table 3.1 shows the top 5 selected SNPs. As would be expected, APOE, one of the top genetic risk factors for AD Liu *et al.* (2013), is ranked #1 for three groups of trials: the diagnose, the volume of hippocampal and the volume of entorhinal cortex.

3.5 Summary

I propose a novel distributed framework for collaborative analyses across institutions while protecting individual data privacy. Safe screening rules are integrated into our framework to identify irrelevant features and speed up the learning process. Experiments demonstrate the efficiency of the proposed framework. D-EDPP achieved a speedup of approximate 66 folds compared to the original solver; I identified several risk genetics factors that associated with AD via stability selection. In the next chapter, I extend this distributed framework to investigate group feature discovery and feature selections across multiple institutions.

Chapter 4

LARGE-SCALE FEATURE SELECTION OF RISK GENETIC FACTORS FOR ALZHEIMER'S DISEASE VIA DISTRIBUTED GROUP LASSO

4.1 Introduction

In this chapter, I propose a distributed feature selection framework to conduct the large-scale imaging genetics studies across multiple institutions. I focus on selecting the relevant group features of ADNI data set, which is associated with optimizing the group Lasso problem Yuan and Lin (2006) in a distributed manner. The learning process is conducted among multiple research institutions without compromising the data privacy for each participating institution.

Collaborative imaging genetics studies across different research institutions show the effectiveness of detecting genetic risk factors. However, the high dimensionality of GWAS data poses significant challenges in detecting risk SNPs for AD. However, processing and integrating genetic data across different institutions is challenging. The first issue is the data privacy since each participating institution wishes to collaborate with others without revealing its own data set. The second issue is how to conduct the learning process across different institutions. Local Query Model (LQM) Li *et al.* (2016b); Zhu *et al.* (2017) is proposed to perform the distributed Lasso regression for large-scale collaborative imaging genetics studies across different institutions while preserving the data privacy for each of them. However, in some imaging genetics studies Harold *et al.* (2009), we are more interested in finding important explanatory factors in predicting responses, where each explanatory factor is represented by a group of features since lots of AD genes are continuous or relative with some other

features, not individual features. In such cases, the selection of important features corresponds to the selection of groups of features. As an extension of Lasso, group Lasso Yuan and Lin (2006) has been proposed for feature selection in a group level and quite a few efficient algorithms Qin *et al.* (2013); Boyd *et al.* (2011) have been proposed for efficient optimization. However, integrating group Lasso with imaging genetics studies across multiple institutions has not been studied well.

In this study, I propose a novel Distributed Feature Selection Framework (DFSF) to conduct the large-scale imaging genetics studies analysis across multiple research institutions. Our framework has three components. In the first stage, I proposed a family of distributed group lasso screening rules (DSR and DDPP_GL) to identify inactive features and remove them from the optimization. The second stage is to perform the group lasso feature selection process in a distributed manner, selecting the top relevant group features for all the institutions. Finally, each institution obtains the learnt model and perform the stability selection to rank the top risk genes for AD. The experiment is conducted on the Alzheimer's Disease Neuroimaging Initiative (ADNI) GWAS data set, including approximately 809 subjects with 5.9 million loci. Empirical studies demonstrate that proposed method the proposed method achieved a 35-fold speedup compared to state-of-the-art distributed solvers like ADMM. Stability selection results show that the proposed DFSF detects *APOE*, *GRM8*, *GPC6* and *LOC100506272* as top risk SNPs associated with AD, demonstrating a superior result compared to Lasso regression methods Li *et al.* (2016b). The proposed method offers a powerful feature selection tool to study AD and its early symptom.

4.2 Problem Statement

Problem Formulation

Group Lasso Yuan and Lin (2006) is a highly efficient feature selection and regression technique used in the model construction. Group Lasso takes the form of the equation:

$$\min_{x \in \mathbb{R}^N} F(x) = \frac{1}{2}||y - \sum_{g=1}^{G}[A]_g[x]_g||_2^2 + \lambda \sum_{g=1}^{G} w_g||[x]_g||_2, \tag{4.1}$$

where A represents the feature matrix where $A \in \mathbb{R}^{N \times P}$ and y denotes the N dimensional response vector. λ is a positive regularization parameter. Different from the Lasso regression problem Tibshirani (1996), group Lasso partitions the original feature matrix X into G non-overlapping groups $[A]_1, [A]_2, ..., [A]_G$ and w_g denotes the weight for the g-th group. After solving the group Lasso problem, I get the corresponding G solution vector $[x]_1, [x]_2, ..., [x]_G$ and the dimension of $[x]_g$ is the same as the feature space in the corresponding design matrix $[A]_g$.

ADNI GWAS Data Set

The ADNI GWAS dataset contains genotype information of 809 ADNI participants. To store statistically relevant SNPs called using Illuminas CASAVA SNP Caller, the ADNI WGS SNP data is stored in variant call format (VCF) for storing gene sequence variations. SNPs at approximately 5.9 million specific loci are recorded for each participant. I encode SNPs using the coding scheme in Sasieni (1997) and apply Minor Allele Frequency (MAF) < 0.05 and Genotype Quality (GQ) < 45 as two quality control criteria to filter high quality SNPs features. I follow the same SNP genotype coding and quality control scheme in Li *et al.* (2016b).

I have m institutions to conduct the collaborative learning. The ith institution

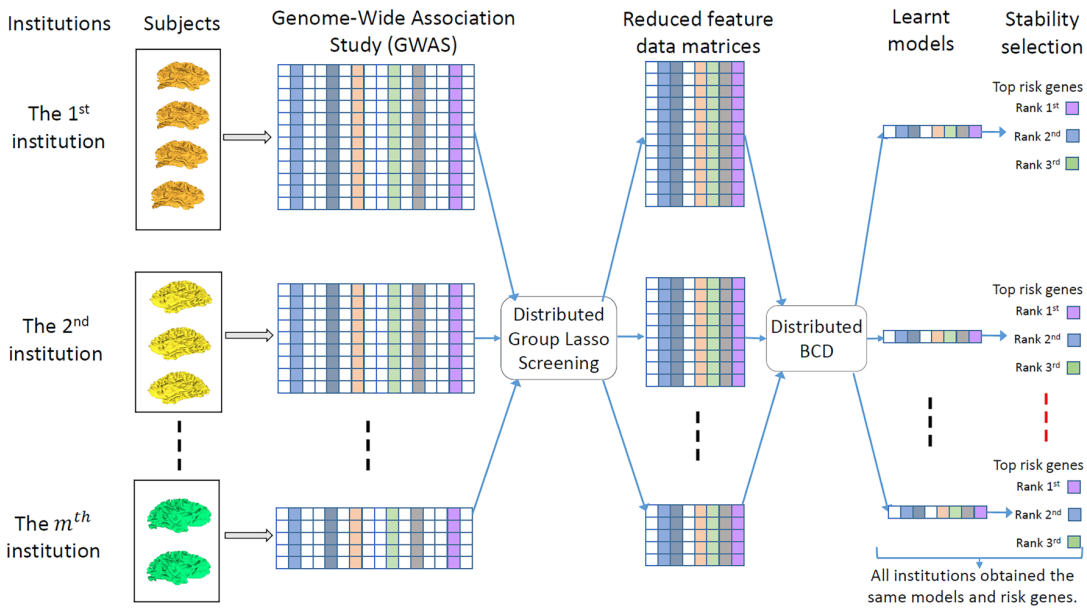


Figure 4.1: The Proposed Feature Selection Framework

maintains its own data set (A_i, y_i) where $A_i \in \mathbb{R}^{n_i \times P}$, n_i is the sample number, P is the feature number and $y_i \in \mathbb{R}^{n_i}$ is the response and $N = \sum_i^m n_i$. I assume P is the same across m institutions. I aim at conducting the feature selection process of group lasso on the distributed datasets (A_i, y_i), $i = 1, 2, ..., m$.

4.3 Proposed Framework

In this section, I present the streamline of proposed DFSF framework. The DFSF framework is composed of three main procedures:

1. Identify the inactive features by the distributed group Lasso screening rules and remove inactive features from optimization.

2. Solve the group Lasso problem on the reduced feature matrix along a sequence of parameter values and select the most relevant group features for each participating institution.

3. Perform the stability selection to rank SNPs that may collectively affect AD.

I Illustrate the framework of our proposed DFSF in Fig 4.1. Each participating institution maintains its own dataset which are a few subjects of GWAS dataset. Firstly, I perform the distributed group Lasso screening rules to pre-identifying the inactive features and remove them from the optimization. Then I conduct the learning process of group Lasso by proposed distributed solver DBCD to select the top relevant features. Finally, each institution obtains the same selected features and performs stability selection to rank the top SNPs that may collectively affect AD. The whole learning procedure results in the same model for all the institutions.

Screening Rules for Group Lasso

Strong rule Tibshirani *et al.* (2012b) is an efficient screening method for fitting lasso-like problems by pre-identifying the features which have zero coefficients in the solution and removing these features from optimization, significantly cutting down on the computation required for optimization.

For the group lasso problem Yuan and Lin (2006), the gth group of x—$[x]_g$— will be discarded by strong rules if the following rule holds:

$$||[A]_g^T y||_2 \leq w_g(2\lambda - \lambda_{\max}) \tag{4.2}$$

The calculation of $\lambda_{\max}$ follows $\lambda_{\max} = \max_g \frac{||[A]_g^T y||_2}{w_g}$. $[x]_g$ could be discarded in the optimization without sacrificing the accuracy since all the elements of $[x]_g$ are zero in the optimal solution vector.

Let J denote the index set of groups in the feature space and $J = \{1, 2, ..., G\}$. Suppose that there are $\widetilde{G}$ remaining groups after employing screening rules, I use $\widetilde{J}$ to represent the index set of remaining groups and $\widetilde{J} = \{1, 2, ..., \widetilde{G}\}$. As a result,

the optimization of group lasso problem (4.1) can be reformulated as:

$$\min_{\widetilde{x}\in\mathbb{R}^{\widetilde{N}}} F(\widetilde{x}) = \frac{1}{2}||y - \sum_{g=1}^{\widetilde{G}}[\widetilde{A}]_g[\widetilde{x}]_g||_2^2 + \lambda\sum_{g=1}^{\widetilde{G}} w_g||[\widetilde{x}]_g||_2, \tag{4.3}$$

where $\widetilde{N}$ is the dimension of reduced feature space and $\widetilde{x} \in \mathbb{R}^{\widetilde{N}}$.

Distributed Screening Rules for Group Lasso

As the data set are distributed among multiple research institutions, it is necessary to conduct a distributed learning process without compromising the data privacy for each institution. LQM Li *et al.* (2016b); Zhu *et al.* (2017) is proposed to optimize the lasso regression while preserving the data privacy for each participating institution. In this study, I aim at selecting the group features to detect the top risk genetic factors for the entire GWAS data set. Since each institution maintains its own data pair (A_i, y_i), I develop a family of distributed group Lasso screening to identify and discard the inactive features in a distributed environment. I summarize the proposed Distributed Strong Rules (DSR) as follows:

1. For the ith institution, compute Q_i by $Q_i = A_i^T y_i$.

2. Update $Q = \sum_i^m Q_i$ by LQM, then send Q back to all the institutions.

3. In each institution, calculate $\lambda_{\max}$ by: $\lambda_{\max} = \max_g \frac{||[Q]_g||_2}{w_g}$ where $[Q]_g$ is the elements of gth group in Q and it is similar as the definition of $[A]_g$.

4. For each gth group in the problem (2.1), I will discard it and remove from the optimization when the following rule holds: $||[Q]_g||_2 \leq w_g(2\lambda - \lambda_{\max})$.

In many real word applications, the optimal value of regularization parameter λ is unknown. To tune the value of λ, commonly used methods such as cross validation needs to solve the Lasso problem along a sequence of parameter values $\lambda_0 > \lambda_1 >$

... $> \lambda_\kappa$,which can be very time-consuming. A sequential version of strong rules was proposed in Tibshirani *et al.* (2012b) by utilizing the information of optimal solutions in the previous parameter. Suppose I have already obtained the solution vector $x(\lambda_{k-1})^*$ at λ_{k-1} where the integer $k \in [1, \kappa]$, the sequential strong rule rejects the ith feature at λ_k when the following rule holds:

$$||[A]_g^T(y - \sum_{g=1}^{G}[A]_g[x]_g(\lambda_{k-1}))||_2 \leq w_g(2\lambda - \lambda_{\max}). \tag{4.4}$$

Strongs rule is one of heuristic screening rules which is able to achieve a substantial speedup by removing inactive features from optimization. However, heuristic screening rules can not guarantee that discarded features have zero components in the solution. In other words, it might mistakenly discard active features from optimization. Tibshirani *et al.* (2012b) solves it by checking KKT conditions to guarantee the correctness of predicted results.

A sequential version of safe screening rules was proposed in EDPP Wang *et al.* (2013a) by utilizing the information of optimal solutions in the previous parameter, achieving about 200x speedups for real-world applications. The implementation details of EDPP is available on the GitHub [1] . I omit the introduction of EPDD for brevity. I propose a distributed safe screening rules for group Lasso, known as the Distributed Dual Polytope Projection Group Lasso (DDPP_GL), to quickly identify and discard the inactive features along a sequence of parameters in a distributed manner. I summarize the proposed DDPP_GL in the algorithm 5.

Distributed Block Coordinate Descent for Group Lasso

After I apply DDPP_GL to discard the inactive features, the feature space shrank from P to $\widetilde{P}$ and there are remaining $\widetilde{G}$ groups. The problem of group Lasso (4.1)

[1] http://dpc-screening.github.io/glasso.html

Algorithm 5 Distributed Dual Polytope Projection for Group Lasso

Require: A set of data pairs $\{(A_1, y_1), (A_2, y_2), ..., (A_m, y_m)\}$ and ith institution holds the data pair (A_i, y_i). A sequence of parameters: $\lambda_{\max} = \lambda_0 > ... > \lambda_\kappa$.

Ensure: The learnt models: $\{x^*(\lambda_0), x^*(\lambda_1), ..., x^*(\lambda_\kappa)\}$.

Let $R_i = A_i^T y_i$, compute $R = \sum_{i=1}^m R_i$ by LQM.

$\lambda_{max} = \max_g \frac{||[R]_g||_2}{w_g}$, $[R]_g$ represents all the elements in the gth group.

$S_i = argmax_{[A_i]_g} \frac{||R_g||_2}{w_g}$, compute $L = \sum_{i=1}^m S_i^T y_i$ by LQM.

Let $\lambda_0 \in (0, \lambda_{\max}]$ and $\lambda \in (0, \lambda_0]$.

$$\theta_i(\lambda) = \begin{cases} \frac{y_i - \sum_{g=1}^G [A_i]_g [x^*(\lambda)]_g}{\lambda}, & \text{if } \lambda \in (0, \lambda_{max}). \\ \frac{y_i}{\lambda_{\max}}, & \text{if } \lambda = \lambda_{max}. \end{cases}$$

$$v_1(\lambda_0)_i = \begin{cases} \frac{y_i}{\lambda_0} - \theta_i(\lambda_0), & \text{if } \lambda \in (0, \lambda_{max}), \\ S_i L, & \text{if } \lambda = \lambda_{max} \end{cases}$$

$v_2(\lambda, \lambda_0)_i = \frac{y_i}{\lambda} - \theta_i(\lambda_0)$

$Q_i = ||v_1(\lambda_0)_i||_2^2$, compute $Q = \sum_i^m Q_i$ by LQM.

$v_2^\perp(\lambda, \lambda_0)_i = v_2(\lambda, \lambda_0)_i - \frac{<v_1(\lambda_0)_i, v_2(\lambda, \lambda_0)_i>}{Q} v_1(\lambda_0)_i$

Given a sequence of parameters $\lambda_{\max} = \lambda_0 > ... > \lambda_\kappa$, for any integer $k \in [1, \kappa]$, I make a pre-screen on each groups of $[x^*(\lambda_k)]_g$, if $[x^*(\lambda_{k-1})]_g$ is known.

for $g = 1$ **to** G **do**

$Q_i = [A_i]_g^T (\theta^*(\lambda_{k-1})_i + \frac{1}{2} v_2^\perp(\lambda_k, \lambda_{k-1})_i)||_2$

Compute $Q = \sum_i^m Q_i$ by LQM.

if $Q < 1 - \frac{1}{2}||v_2^\perp(\lambda_k, \lambda_{k-1})||_2 ||[A]_g||_2$ **then**

I identify all the elements of $[x^*(\lambda_k)]_g$ to be zero.

end for

could be reduced as (4.3). I need to optimize (4.3) in a distributed manner. The block coordinate descent (BCD) Qin *et al.* (2013) is one of the most efficient solvers in the big data optimization. BCD optimize the problem by updating one or a few blocks of variables at a time, rather than updating all the block together. The order of update can be a deterministic or stochastic sequence. For the group lasso problem, I can randomly pick up a group of variables to optimize and keeps other groups of variables fixed. Following this idea, I propose a Distributed Block Coordinate Descent (DBCD) to optimize the group Lasso problem in the algorithm 6 via a distributed manner.

In algorithm 6, I use a variable R_i to store the result of $\widetilde{A}_i\widetilde{x} - y_i$. R_i is initialized as $-y_i$ since $\widetilde{x}$ is initialized to be zero at the beginning. In DBCD, the update of gradient can be divided as three steps:

Step 1: Compute the gradient: $\nabla F([\widetilde{x}]_g)_i = [\widetilde{A}_i]_g^T R_i$ and get $\nabla F([\widetilde{x}]_g)$ by LQM.

Step 2: Get $\Delta[\widetilde{x}]_g$ by the gradient information $\nabla F([\widetilde{x}]_g)$.

Step 3: Update R_i: $R_i = R_i + \Delta[\widetilde{x}]_g[\widetilde{A}_i]_g^T$.

The update of $[\widetilde{x}]_g$ follow the equations in 7rd line of algorithm 6. I update $[\widetilde{x}]_g$ if $||[\widetilde{x}]_g||_2$ is larger than $\frac{\lambda w_g}{L_g}$, otherwise all the elements of $[\widetilde{x}]_g$ are set to be zero. L_g denotes the Lipschitz constant in gth group. For the group Lasso problem, L_g is set to be $||[A]_g||_2^2$. DBCD updates R_i at the end of each iteration to make sure R_i stores the correct information of $\widetilde{A}_i\widetilde{x} - y_i$ in each iteration.

Feature Selection by Group Lasso

Given a sequence of parameter values: $\lambda_0 > ... > \lambda_\kappa$, I can obtain a sequence of learnt models $\{x^*(\lambda_0), ..., x^*(\lambda_\kappa)\}$ by employing DDPP_GL+DBCD. For each group g in the feature space G, I count the frequency of nonzero entries in the learnt model

Algorithm 6 Distributed Block Coordinate Descent

Require: A set of data pairs $\{(\widetilde{A}_1, y_1), (\widetilde{A}_2, y_2), ..., (\widetilde{A}_n, y_n)\}$ where ith institution holds the data pair $(\widetilde{A}_i, y_i)$ and λ

Ensure: The learnt models: $\widetilde{x}$.

Initialize: $\widetilde{x} = \mathbf{0} \in \mathbb{R}^{\widetilde{P}}$ and $R_i = -y_i$.

while not converged **do**

Randomly pick up g from the index set $\{1, ..., \widetilde{G}\}$.

Compute the gth group's gradient: $\nabla F([\widetilde{x}]_g)_i = [\widetilde{A}_i]_g^T R_i$.

Update $\nabla F([\widetilde{x}]_g)$ by LQM: $\nabla F([\widetilde{x}]_g) = \sum_i^m \nabla F([\widetilde{x}]_g)_i$.

Let $v = [\widetilde{x}]_g$ and $[\widetilde{x}]_g = [\widetilde{x}]_g - \frac{1}{L_g} \nabla F([\widetilde{x}]_g)$

$$[\widetilde{x}]_g = \begin{cases} [\widetilde{x}]_g - \frac{\lambda w_g}{||[\widetilde{x}]_g||_2}[\widetilde{x}]_g, & \text{if } ||[\widetilde{x}]_g||_2 > \frac{\lambda w_g}{L_g}. \\ \mathbf{0} \in \mathbb{R}^{\widetilde{N}}, & \text{if } ||[\widetilde{x}]_g||_2 \leq \frac{\lambda w_g}{L_g}. \end{cases}$$

Let compute $\Delta[\widetilde{x}]_g$ by: $\Delta[\widetilde{x}]_g = [\widetilde{x}]_g - v$.

Update R_i by: $R_i = R_i + \Delta[\widetilde{x}]_g [\widetilde{A}_i]_g^T$

end while

and rank the frequency by descent to get the top relevant features. I summarize the top K feature selection process as follows:

1. For each group g in the feature space G, $I_g = I_g + 1$, If $[x^*(\lambda_k)]_g$ is not equal to zero where $k \in (0, \kappa)$ and $I \in \mathbb{R}^G$.

2. Rank I by descent and select the top K relevant features from A_i to construct the feature matrix $\bar{A}_i$.

After obtaining the relevant features, I perform the stability selection Li *et al.* (2016b); Meinshausen and Buhlmann (2010) to rank the top genetic factors that are associated with the disease AD.

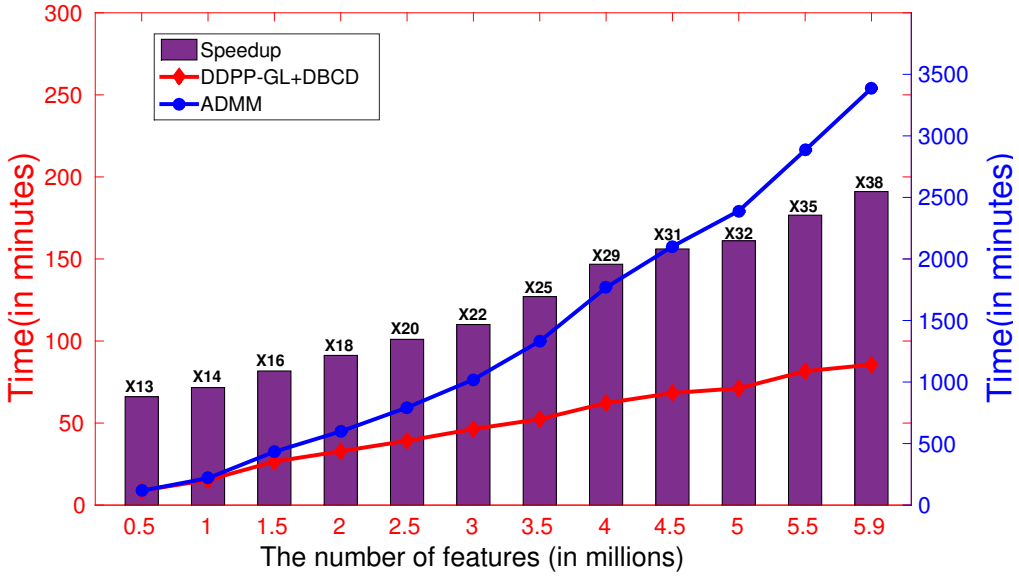


Figure 4.2: The Running Time Comparison of DDPP_GL+DBCD with the distributed ADMM.

4.4 Experimental Results

In this section, I conduct several experiments to evaluate the efficiency and effectiveness of our methods. The proposed framework is implemented across three institutions with thirty computation nodes on Apache Spark [2] , a state-of-the-art distributed computing platform. I perform DDPP_GL+ DBCD on a sequence of parameter values and employ stability selection with our methods to determine top risk SNPs related to AD.

Performance Comparison

In this experiment, I choose the volume of lateral ventricle as variables being predicted which containing 717 subjects by removing subjects without labels. The volumes of brain regions were extracted from each subject's T1 MRI scan using Freesurfer

[2]http://spark.apache.org

[3] . The distributed platform is built across three research institutions that maintain 326, 215, and 176 subjects, respectively and each institution has ten computation nodes. I perform the DDPP_GL+DBCD along a sequence of 100 parameter values equally spaced on the linear scale of $\lambda/\lambda_{\max}$ from 1.00 to 0.1. As a comparison, I run the state-of-the-art distributed solver ADMM Boyd *et al.* (2011) with the same experiment setup. The group size is set to be 20 and I vary the number of features by randomly selecting 0.5 million to 5.9 million from GWAS dataset and report the result in Fig 4.2. The proposed method achieved a 38-fold speedup compared to ADMM.

Stability Selection for Top Risk Genetic Factors

I employ stability selection Li *et al.* (2016b); Meinshausen and Buhlmann (2010) with DDPP_GL+DBCD to select top risk SNPs from the entire GWAS data set with 5,906,152 features. I conduct two different groups of trials by choosing the volume of hippocampus and entorhinal cortex at baseline as the response variable for each group, respectively. In each trial, DDPP_GL+DBCD is carried out along a 100 linear-scale sequence of parameter values from 1 to 0.05, respectively. Then I select the top 10000 features and perform stability selection Meinshausen and Buhlmann (2010) to rank the top risk SNPs for AD. As a comparison, I perform D_EDPP+F_LQM Li *et al.* (2016b) with the same environment setup and report the result in Table 4.1. In the trials, *APOE* is ranked 1st by both of D_EDPP+F_LQM and DDPP_GL+DBCD. However, DDPP_GL+DBCD detects more risk genes like *GRM8*, *GPC6*, *PIK3C2G* and *LOC100506272* that are associated with the disease AD in GWAS Rouillard *et al.* (2016).

[3]http://freesurfer.net

Table 4.1: Top 5 Selected SNPs with the Volume of Entorhinal Cortex and Hippocampal.

Hippocampus by D_EDPP+F_LQM				Hippocampus by DDPP_GL+DBCD			
No.	Chr	RS_ID	Gene	No.	Chr	RS_ID	Gene
1	19	rs429358	APOE	1	19	rs429358	APOE
2	8	rs34173062	SHARPIN	2	7	rs1592376	GRM8
3	6	rs71573413	unknown	3	5	rs6892867	LOC105377696
4	11	rs10831576	GALNT18	4	6	rs71573413	unknown
5	9	rs3010760	unknown	5	13	rs7317246	GPC6
Entorhinal by D_EDPP+F_LQM				Entorhinal by DDPP_GL+DBCD			
No.	Chr	RS_ID	Gene	No.	Chr	RS_ID	Gene
1	19	rs429358	APOE	1	19	rs429358	APOE
2	15	rs8025377	ABHD2	2	4	rs1876071	LOC100506272
3	Y	rs79584829	unknown	3	18	rs4486982	unknown
4	14	rs41354245	MDGA2	4	14	rs41354245	MDGA2
5	3	rs55904134	unknown	5	12	rs12581078	PIK3C2G

4.5 Summary

I propose a novel distributed feature selection framework to select the relevant group features for the study of risk genetics factors of AD. The feature selection framework is based on optimizing the distributed group Lasso problem along a sequence of parameter values. The group Lasso screening rules are integrated into our framework to identify irrelevant features and speed up the learning process.

Experiments demonstrate the efficiency of the proposed framework. The proposed DDPP_GL+DBCD achieved a speedup of approximate 38 folds compared to the state-of-the-art distributed solver ADMM Boyd *et al.* (2011). Moreover, the proposed distributed framework identified more risk genetics factors via stability selection.

Chapter 5

SCALING UP THE DICTIONARY LEARNING AND SPARSE CODING BY STOCHASTIC COORDINATE CODING

5.1 Introduction

Sparse dictionary learning is a representation learning method which aims at finding a sparse representation of the input data (also known as sparse coding) in the form of a linear combination of basic elements as well as those basic elements themselves. Dictionary learning and sparse representation methodology developed in the machine learning field has been shown to be efficient in learning adaptive, over-complete and diverse features for optimal representations. Despite the rich promise of sparse coding models, sparse coding is computationally expensive especially when dealing with large-scale data. The main computational cost of sparse coding lies in the updating of sparse codes and the dictionary. It is known that updating the sparse code is usually much more time consuming. Therefore, much of recent work has been devoted to seeking efficient optimization algorithms for updating the sparse code Lee *et al.* (2007); Tibshirani *et al.* (2012a). The basic idea of these methods is to quickly identify the non-zero entries of the sparse code, thus reducing the search space. However, most of these algorithms are iterative batch methods which may not scale to very large data sets Bousquet and Bottou (2008), since updating the dictionary involves the computation of the full gradient of the dictionary from the whole data set and is expensive. Solving sparse coding remains a computationally challenging problem, especially when dealing with large-scale data sets and learning large size dictionaries. In this study, I propose a novel algorithm to solve the dictionary learning and

sparse coding problem, called Stochastic Coordinate Coding (SCC). The proposed algorithm alternatively updates the sparse codes via just a few steps of coordinate descent and updates the dictionary via second order stochastic gradient descent. The computational cost is further reduced by focusing on the non-zero components of the sparse codes and the corresponding columns of the dictionary only in the updating procedure. Thus, the proposed algorithm significantly improves the efficiency and the scalability, making sparse coding applicable for large-scale data sets and large dictionary sizes. Our experiments are conducted on the Drosophila gene expression data sets. Drosophila melanogaster has been established as a model organism for investigating the fundamental principles of developmental gene interactions. The gene expression patterns of Drosophila melanogaster can be documented as digital images, which are annotated with anatomical ontology terms to facilitate pattern discovery and comparison. The automated annotation of gene expression pattern images has received increasing attention due to the recent expansion of the image database. The effectiveness of gene expression pattern annotation relies on the quality of feature representation. In this study, I conduct an unsupervised dictionary learning method to learn the features for the Drosophila image database. The experiment demonstrate the efficiency and the effectiveness of the proposed algorithm.

5.2 Problem Formulation

I first introduce our notation used in this study. Given a data set $X = (x_1, ..., x_n)$ of image patches, each image patch is a p-dimensional vector, $x_i \in \mathbb{R}^p, i = 1, ..., n$. Moreover, each x_i is preprocessed to be zero mean and unit L_2 norm. I first extract meaningful features from these image patches using sparse coding. The learned features will be used for image annotation. The linear decomposition of an image patch using a few number of basis or atoms of a learned dictionary has recently led to

state-of-art performance in numerous signal processing and machine learning tasks. Specifically, suppose there are m atoms $d_j \in \mathbb{R}^p, j = 1, ..., m$, where the number of atoms is usually much smaller than the number of image patches n but larger than the dimension of the image patch p. Each image patch can then be represented as $x_i = \sum_{j=1}^{m} z_{i,j} d_j$. Therefore, each p-dimensional image patch x_i is represented by a m-dimensional vector $z_i = (z_{i,1}, ..., z_{i,m})$. It is further assumed that each image patch can be represented only by a small group of atoms, that is, the learned feature vector z_i is a sparse vector.

Given one image patch x_i, one can formularize the above idea as the following optimization problem:

$$\min f_i(D, z_i) = \frac{1}{2}||x_i - Dz_i||_2^2 + \lambda||z_i||_1, \tag{5.1}$$

where λ is the regularization parameter, $||.||$is the standard Euclidean norm and $||z_i||_1 = \sum_{j=1}^{m} |z_{i,j}|$. The first term of Eq.(5.1) is the reconstruction error, which measures how well the new feature represents the image patch. The second term of Eq.(5.1) ensures the sparsity of the learned feature z_i. Each z_i is often called the *sparse code*. Since z_i is sparse, there are only a few entries in z_i which are non-zero. I call its non-zero entries as its *support*, supp $(z_i) = \{z_{i,j} : z_{i,j} \neq 0, j = 1, ..., m\}$. Here $D = (d_1, ..., d_m) \in \mathbb{R}^{m \times p}$ is called the dictionary. To prevent an arbitrary scaling of the sparse code, each column of D is restricted to be in a unit ball, $d_j \leq 1$. Given the whole data set $X = (x_1, ..., x_n)$, the sparse coding problem is then given as follows:

$$\min_{D \in \psi, z_1, ..., z_m} F(D, z_1, ..., z_m) = \frac{1}{n} \sum_{i=1}^{n} f_i(D, z_i), \tag{5.2}$$

where ψ is the feasible set of D which is defined as follows:

$$D = \{D \in \mathbb{R}^{p \times m} : \forall j = 1, ..., m, ||d_j||_2 \leq 1\}. \tag{5.3}$$

It is a non-convex problem with respect to joint parameters in the dictionary D and the sparse codes $Z = (z_1, ..., z_n)$. Therefore, it is often difficult to find a global

optimum. However, it is a convex problem when either D or Z is fixed. When the dictionary D is fixed, solving each sparse code z_i is the well known lasso problem. Many methods have been proposed to solve this problem, including Least Angle Regression (LARS) Efron *et al.* (2004), Fast Iterative Soft-Thresholding Algorithm (FISTA) Beck and Teboulle (2009) and Coordinate Descent (CD) Shalev-Shwartz and Tewari (2011). It might be worth noting that when the feature dimension m is large which is often the case, solving a lasso problem is very time consuming. When the sparse codes are fixed, it is a simple quadratic problem. Therefore, one often uses an alternating optimization approach to solve the sparse coding problem. Specifically, when D is fixed, I update the sparse code z_i for each image patch x_i. When the sparse codes are fixed, I use gradient descent to update the dictionary:

$$D \leftarrow D - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla_D f_i(D, z_i) = D - \eta \frac{1}{n} \sum_{i=1}^{n} (Dz_i - x_i) z_i^T, \tag{5.4}$$

where η is the step size. However, at each iteration, full gradient descent requires evaluation of n derivatives, which is very expensive when the data set is of large-scale. A popular modification is Stochastic Gradient Descent (SGD) Zhang (2004). At each iteration, I randomly draw an image patch x_t, and update the dictionary as follows:

$$D_{t+1} \leftarrow D_t - \eta_t \nabla_{D_t} f_t(D_t, z_t), \tag{5.5}$$

where η_t is called the learning rate.

I summarize the optimization methods in the following. First I initialize the dictionary D. Many dictionary initialization methods have been proposed, such as random weights Jarrett *et al.* (2009), random patches and k-means. A detailed comparison of the performance among these initialization methods has been discussed in Coates and Ng (2011). With the initial dictionary, conventional sparse coding algorithms

include the following main steps:

Step 1: Get an image patch x_i.

Step 2: Calculate the sparse code z_i by using LARS, FISTA or coordinate descent.

Step 3: Update the dictionary D by performing stochastic gradient descent.

Step 4: Go to step 1 and iterate.

I call each cycle, each image patch has been trained once, as an epoch. Usually, several epochs are required to obtain a satisfactory result. When the number of image patches and the dictionary size is large, step 2 and step 3 are still very slow. I propose a novel algorithm to improve both of these parts, which is presented in the next section.

5.3 Stochastic Coordinate Coding

In this chapter, I introduce our Stochastic Coordinate Coding (SCC) algorithm. It is known that solving the sparse coding problem usually is very time consuming especially when dealing with large-scale data sets and large size dictionaries Lee *et al.* (2007). The proposed algorithm aims to dramatically reduce the computational cost of the sparse coding while keeping comparable performance.

I detail our algorithm in the following. Initialize the dictionary via any initialization method and denote it as D_1^1. Initialize the sparse code $z_i^0 = 0$ for $i = 1, ..., n$. Here I use superscript to represent the number of epochs and I use subscript to represent the index of data points. Then starting from $k = 1$ and $i = 1$, I do the following:

1. Get an image patch x_i.

2. Update z_i^k via one or a few steps of coordinate descent:

$$z_i^k = CD(D_i^k, z_i^{k-1}, x_i). \tag{5.6}$$

Specifically, for j from 1 to m, I update the j-th coordinate $z_{i,j}^{k-1}$ of z_i^{k-1} cyclically as follows:

$$b_j \leftarrow (d_{i,j}^k)^T(x_i - D_i^k z_i^{k-1}) + z_{i,j}^{k-1}, \tag{5.7}$$

$$z_{i,j}^{k-1} \leftarrow \Gamma_\lambda(b_j), \tag{5.8}$$

where Γ is the soft thresholding shrinkage function Combettes and Wajs (2005). I call such one updating cycle as *one step* of coordinate descent. The updated sparse code is then denoted by the term z_k^i.

3. Update the dictionary D by using stochastic gradient descent:

$$D_{i+1}^k = P_\Psi(D_i^k - \eta_i^k \nabla_{D_i^k} f_i(D_i^k, z_i^k)), \tag{5.9}$$

where P denotes the projection operator. I set the learning rate as an approximation of the inverse of the Hessian matrix. The gradient of D_i^k can be obtained as follows:

$$\nabla_{D_i^k} f_i(D_i^k, z_i^k) = (D_i^k z_i^k - x_i)(z_i^k)^T. \tag{5.10}$$

4. $i = i + 1$. If $i > n$, then set $D_1^{k+1} = D_{n+1}^k$, $k = k + 1$ and $i = 1$.

I illustrate our algorithmic framework in Fig 5.1. At each iteration, I get an image patch x_i. Then I perform one or a few steps of coordinate descent to find the support of the sparse code. Next, I perform a few steps of coordinate descent on the support to obtain a new sparse code z_i^k. Then I update the support of the dictionary by second order stochastic gradient descent.

It is known that the second step - updating the sparse code is the most time consuming part Balasubramanian *et al.* (2013). Coordinate descent is known as one of the state of art methods for solving this lasso problem. Given an image patch x_i, coordinate descent initialize $z_i^0 = 0$ and then update the sparse code many times via

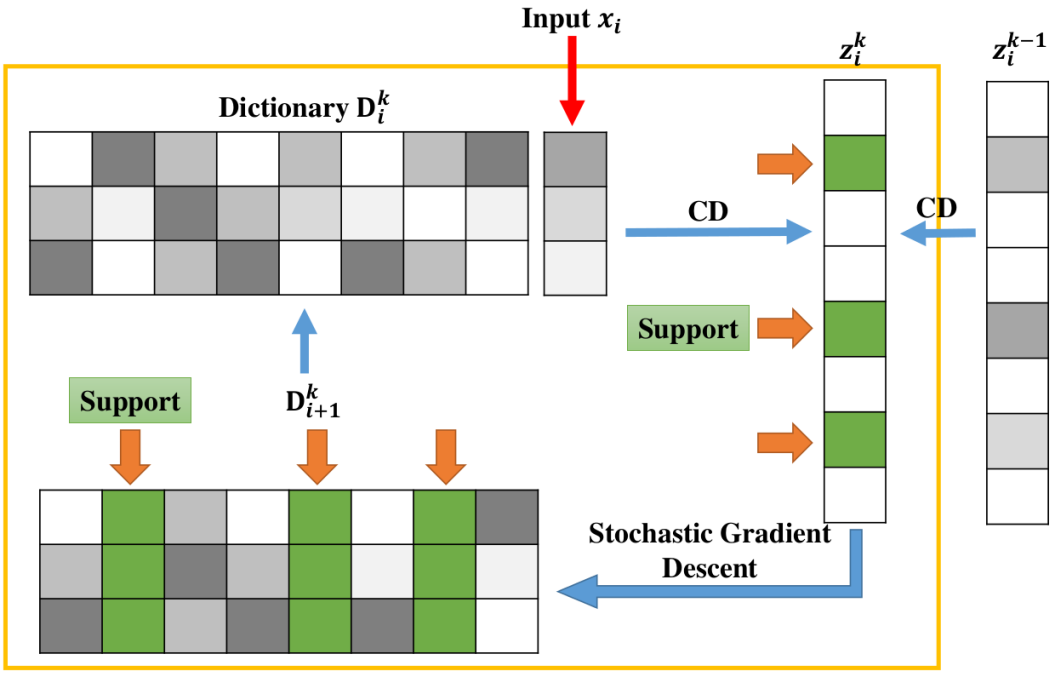


Figure 5.1: Illustration of Our Algorithmic Framework. With an Image Patch x_i, I Perform One Step of Coordinate Descent to Find the Support of the Sparse Code. Next, I Perform a Few Steps of Coordinate Descent on the Support to Obtain a New Sparse Code z_i^k. Then I Update the Support of the Dictionary by Second Order Stochastic Gradient Descent to Obtain a New Dictionary D_{i+1}^k.

matrix-vector multiplication and thresholding. Empirically, iterations may take tens hundreds steps to converge. However, after a few steps, the support of coordinates, the locations of the nonzero entries in z_i, is very accurate, usually less than ten steps. Note that the support of the sparse code is usually more important than the exact value of the sparse code. Moreover, since the original sparse coding is a non-convex problem and it involves an alternating updating, I do not need to run the coordinate descent to final convergence. Therefore, I propose to update the sparse code z_i by using a few steps of coordinate descent. For the k-th epoch, I denote the updated sparse code as z_i^k. It will be used as an initial sparse code for the k+1th epoch.

After updating the sparse code, I know its support. One of my key insights is that when updating the dictionary, I can only need to focus on the support of the dictionary but not all columns of the dictionary. Let $z_{i,j}^k$ denote j-th entry of z_i^k and let $d_{i,j}^k$ denote the j-th column of the dictionary D_i^k. If $z_{i,j}^k = 0$, then $\nabla_{d_{i,j}^k} f_i(D_i^k, z_i^k) = (D_i^k z_i^k - x_i)(z_{i,j}^k)^T = 0$. Therefore, $d_{i,j}^k$ does not need to be updated. Assume $z_{i,j}^k$ is non-zero. Let $d_{i,j}^k$ denote the j-th column of the dictionary D_{i+1}^k. Then I can update $d_{i+1,j}^k$ as follows:

$$d_{i+1,j}^k \leftarrow d_{i,j}^k - \eta_{i,j}^k \nabla_{d_{i,j}^k} f_i(D_i^k, z_i^k) = d_{i,j}^k - \eta_{i,j}^k z_{i,j}(D_i^k z_i^k - x_i), \tag{5.11}$$

Note that z_i^k here is a sparse vector, therefore computing $D_i^k z_i^k$ is very efficient. The computational cost will be significantly reduced when the support is very small. Note that for online dictionary learning, one usually has to update all columns of the dictionary. It is because that online dictionary learning uses the averaged gradient, which is usually not sparse. In other words, the support of the dictionary is itself. Therefore, one has to update all columns of the dictionary for each image patch. It is time consuming especially when the dictionary size is very large.

When the data sets are very large, the learning rate η_i^k will be very small after going through large number of image patches. In this case, the dictionary will not change very much and the efficiency of the training will decrease. In practice, turning the learning rate is very tricky and sensitive. In this study, I use an adaptive learning rate. I aim to design a learning rate with the following two principles. The first one is that for different columns of the dictionary, I may use different learning rates. The second is that for the same column, the learning rate should decrease. Otherwise, the algorithm might not converge. To obtain the learning rate, I use the Hessian matrix of the objective function. It can be shown that the following matrix provides an approximation of the Hessian: $H = \sum_{k,i} z_i^k (z_i^k)^T$, when k and i go to infinity.

Algorithm 7 SCC (Stochastic Coordinate Coding)

Require: Data set $X = (x_1, ..., x_n) \in \mathbb{R}^{p \times n}$.

Ensure: $D \in \mathbb{R}^{p \times m}$ and $Z = (x_1, ..., x_n) \in \mathbb{R}^{m \times n}$.

Initialize: $D_1^1 = 0$, $H = 0$ and $z_i^0 = 0$ for $i = 1, ..., n$.

for $k = 1$ **to** κ **do**

for $i = 1$ **to** n **do**

Get an image patch x_i.

Update z_i^k via one or a few steps of coordinate descent:

$$z_i^k \leftarrow CD(D_i^k, z_i^{k-1}, x_i).$$

Update the Hessian matrix and the learning rate:

$$H \leftarrow H + z_i^k (z_i^k)^T,\ \eta_{i,j}^k = \frac{1}{h_{jj}}.$$

Update the support of the dictionary via SGD:

$$d_{i+1,j}^k \leftarrow d_{i+1,j}^k - \eta_{i,j}^k z_{i,j} (D_i^k z_i^k - x_i).$$

If $i = n$, set $D_1^{k+1} = D_{n+1}^k$.

end for

end for

Output

$D = D_n^\kappa$ and $z_i = z_i^\kappa$ for $i = 1, ..., n$.

According to the second order stochastic gradient descent, I should use the inverse matrix of the Hessian as the learning rate. However, computing a matrix inversion problem is computationally expensive. In order to obtain the learning rate, I simply use the diagonal element of the matrix H. Note that if the columns of the dictionary have low correlation, H is close to a diagonal matrix. Specifically, I first initialize $H = 0$. Then update the matrix H as follows:

$$H \leftarrow H + z_i^k (z_i^k)^T. \tag{5.12}$$

When updating the j-th column for the ith image patch x_i, I replace $\eta_{i,j}^k$ in equation (5.11) by $1/h_{jj}$, where h_{jj} is the j-th diagonal element of H. In this way, I do not have to tune the learning rate parameter. It might be worth noting that I do not have to store the whole matrix of H but only its diagonal elements. I summarize our algorithm in Algorithm 7.

5.4 Experiments

In this chapter, I empirically evaluate the efficiency and effectiveness of our proposed Stochastic Coordinate Coding (SCC) algorithm. The code of SCC is available on the GitHub [1] . A detailed description of data and experimental setting is given in Chapter 5.4.1. I study the influence of different settings of SCC in Chapter 5.4.2, including the influence of the number of coordinate descent steps and the learning rate. Finally, I compare SCC with the state-of-art sparse coding algorithm - Online Dictionary Learning (ODL) Mairal *et al.* (2009) in terms of speedup and accuracy.

Data Description and Experimental Setting

The Drosophila gene expression images used in our work are obtained from the FlyExpress database, which contains standardized images from the Berkeley Drosophila Genome Project (BDGP). The Drosophila embryogenesis is partitioned into 6 stage ranges (1-3, 4-6, 7-8, 9-10, 11-12, 13-17) in BDGP. I focus on the later 5 stage ranges as there are few keywords appeared in the first stage range.

The Drosophila embryos are 3D objects Weber *et al.* (2009), and the FlyExpress database contains 2D images that are taken from different views (lateral, dorsal, and lateral-dorsal) Mace *et al.* (2010). As majority of images in the database are in lateral view Ji *et al.* (2009), I focus on the lateral-view images in our study. For each image,

[1] https://github.com/liohzhee/Stochastic-Coordinate-Coding

Table 5.1: The Comparison of Computational Time Between SCC and ODL for Different Dictionary Sizes

Methods	ODL			SCC		
Dictionary Sizes	500	1000	2000	500	1000	2000
Update Z	2.58	8.83	38.75	0.22	0.39	0.71
Update D	5.23	19.48	63.89	0.03	0.03	0.04
Total time (in hours)	7.81	28.31	102.64	0.25	0.42	0.75

I first use a 16×16 window to obtain a collection of small image patches. Then I extract a 128-dimensional Scale-Invariant Feature Transform (SIFT) Lowe (1999) feature from each image patch. Each SIFT feature is further normalized to be zero mean and unit L_2 norm. The patches with small standard deviations were discarded. After preprocessing the data, I have 555009, 259882, 286349, 989653, 1006012 image patches for different stage ranges (4-6, 7-8, 9-10, 11-12, 13-17) respectively.

For each state range, I first initialize the dictionary via selecting random patches Coates and Ng (2011), which has been shown to be a very efficient and effective initialization method in practice. Then I learn the sparse codes by different sparse coding methods using the same initial dictionary. All 5 stage ranges are trained for 10 epochs using a batch size of 1. After learning the sparse codes, I apply max pooling Scherer *et al.* (2010) to generate the features for annotation. Finally, I employ the one-against-rest support vector machines (SVM) Chang and Lin (2011) to annotate the gene expression pattern images.

The number of Coordinate Descent Steps

In this experiment, I study the influence of the number of coordinate descent steps to the convergence. I use the state range 2 in our experiments. It has 555009 image patches and each image patch is of 128 dimensions. The dictionary size is 1000×128 . I tested 1, 3, 5, 7, 9 steps of coordinate descent. The results are evaluated by the objective function value and the running time, as shown in Fig 5.2.

It can be seen from Fig 5.2 that using a great number of coordinate descent steps can achieve lower objective function value, however the computational time would increase. Therefore I should choose a suitable number of coordinate descent steps. In practice, I choose 3 steps of coordinate descent, which performs quite well in all experiments. Also, I can see from the left figure of Fig 5.2 that SCC converges under coordinate descent steps.

Computational Time Comparison

I first show the computational time of updating the dictionary and updating the sparse code. Table 5.1 shows the computational time of these two steps on three different dictionary sizes, 500×128, 1000×128 and 2000×128. It can be seen from the table that SCC significantly reduces the computational time. Note that when the size of the dictionary increases, the computational time of OL increases rapidly. However, for SCC the computational time increases much slower compared to OL, especially the computational time of updating the dictionary. Therefore, SCC has a better scalability when dealing with large size dictionaries.

A computational time as well as an objective function value comparison is given in Table 5.2. It can be seen from the table that SCC archives a very low objective function value, which is comparable with ODL. Meanwhile, the computational time

Table 5.2: A Comparison of SCC and ODL on the Computational Time.

128×500	Stages	4-6	7-8	9-10	11-12	13-17
Objective	ODL	0.137	0.150	0.147	0.147	0.149
function value	SCC	0.138	0.150	0.148	0.147	0.150
Running time	ODL	15.71	7.81	8.60	30.64	31.14
(in hours)	SCC	0.144	0.068	0.075	0.257	0.262
	Speedup	109.10	114.85	114.67	119.22	118.85
128×1000	Stages	4-6	7-8	9-10	11-12	13-17
Objective	ODL	0.132	0.142	0.140	0.140	0.142
function value	SCC	0.132	0.143	0.141	0.141	0.143
Running time	ODL	59.61	28.31	30.77	107.80	111.01
(in hours)	SCC	0.189	0.090	0.098	0.340	0.347
	Speedup	315.39	314.55	313.97	317.05	319.91
128×2000	Stages	4-6	7-8	9-10	11-12	13-17
Objective	ODL	0.127	0.136	0.134	0.134	0.137
function value	SCC	0.128	0.137	0.135	0.135	0.138
Running time	ODL	219.21	102.64	113.09	390.88	397.34
(in hours)	SCC	0.144	0.068	0.075	0.257	0.262
	Speedup	1522.3	1509.4	1507.8	1520.9	1516.5

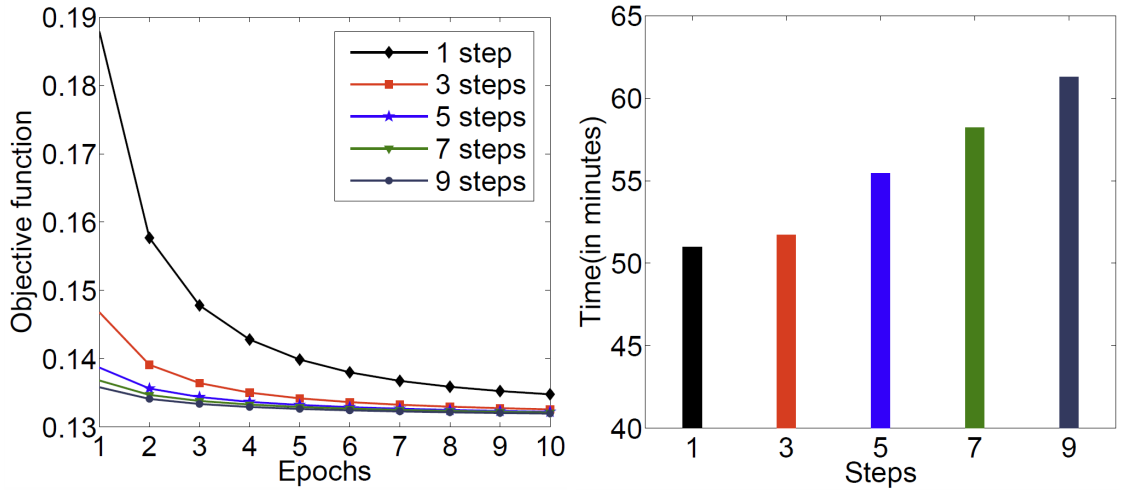


Figure 5.2: A Comparison of Different Coordinate Descent Steps. The Figure on the Left Shows the Objective Value Curves When Varying the Number of Coordinate Descent Steps. The Horizontal Axis Represents the Number of Epochs. The Figure on the Right Shows the Computational Time (in Minutes) of Running 10 Epochs. It Can Be Seen From the Figure that Using a Great Number of Coordinate Descent Steps Can Achieve Lower Objective Value. However, the Overall Computational Time Would Increase.

of SCC is much less than ODL. Note that when the dictionary size increases, the objective function value decreases.

In this work, I focus on the the single batch size setting, that is, I process one image patch in each iteration. I also compare our proposed SCC (with a batch size of 1) with mini-batch ODL (with a batch size of 512). Our empirical results show that the mini-batch ODL is about 3-4 times faster than SCC. Note that the mini-batch algorithms are usually faster than incremental algorithms. In addition, the implementation of mini-batch ODL is well optimized, making a direct time comparison difficult.

Classification Performance Comparison

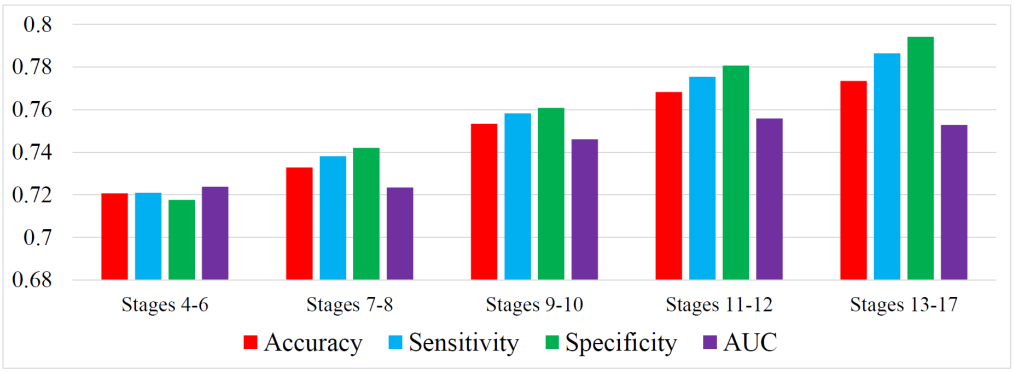
In this experiment, I compare the results of Drosophila gene image annotation by using learned features from SCC and ODL. I tested all 5 stage ranges with different dictionary sizes, i.e., 500×128, 1000×128 and 2000×128. I choose four measurements: accuracy, AUC, sensitivity and specificity to evaluate the performance of different approaches. Comparison results for all 5 stage ranges by a weighted average of the top 10 terms are shown in Fig 5.3 and Fig 5.2, respectively.

It can be seen from the figure that when the dictionary size is 500×128, ODL performs slightly better SCC. When the dictionary size is 1000×128 or 2000×128, SCC and ODL achieve comparable results. However, SCC is significantly faster than ODL in this case. It might be worth noting that when the dictionary size increases, SCC and ODL both improve their performance.

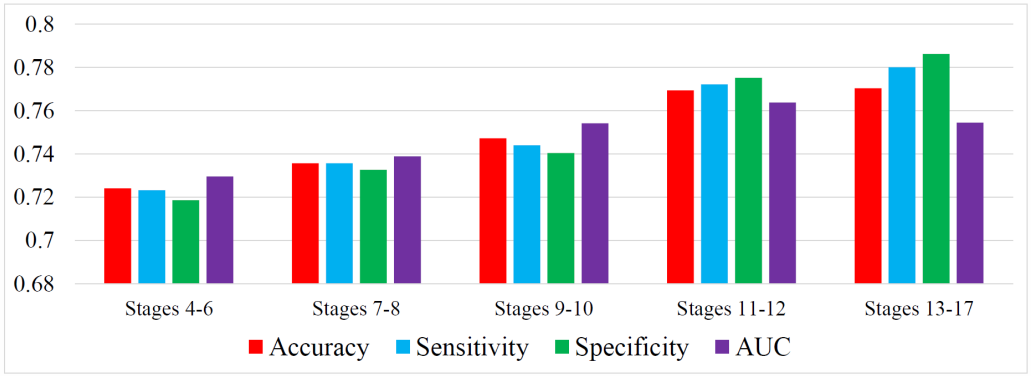
5.5 Summary

In this study, I propose a novel algorithm called Stochastic Coordinate Coding (SCC) to solve the dictionary learning and sparse coding problem. In SCC, I perform a few steps of coordinate descent to update the sparse codes and use second order stochastic gradient descent to update the dictionary. The sparse code is first updated via one step of coordinate descent. Then only the nonzero entries of the sparse code are updated. The computational cost is further reduced by only updating the support of the sparse codes and the dictionary. What's more, I propose an addaptive method to update the learning rate. In SCC, each coordinate of the sparse code has its own learning rate. The learning rates are updated adaptively. Extensive experiments on Drosophila gene expression data sets have demonstrated the efficiency of the proposed algorithm. Compared to the state-of-art sparse coding algorithms, the

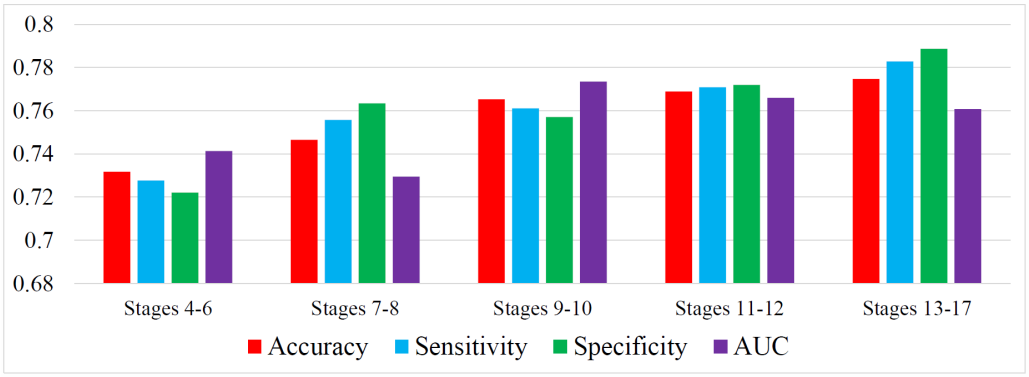
proposed algorithm achieves one or two orders of magnitude speedup when varying the dictionary size. In the next chapter, I extend the proposed dictionary learning method to a multi-task learning framework.



(a) ODL ($m = 500$)

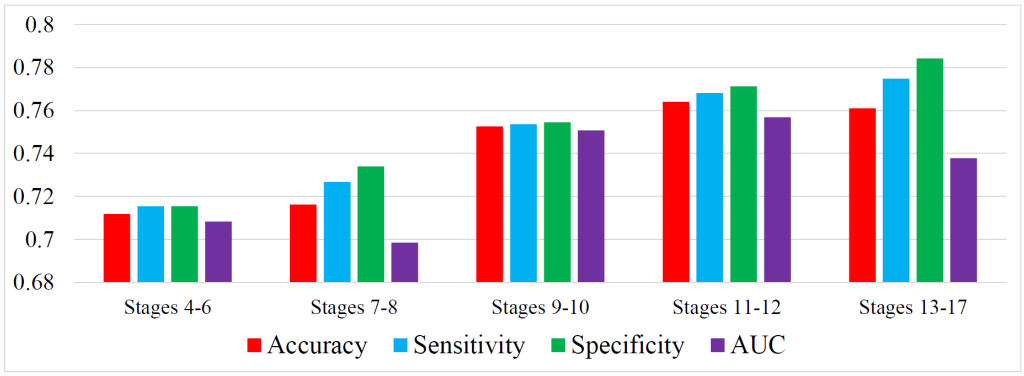


(b) ODL ($m = 1000$)

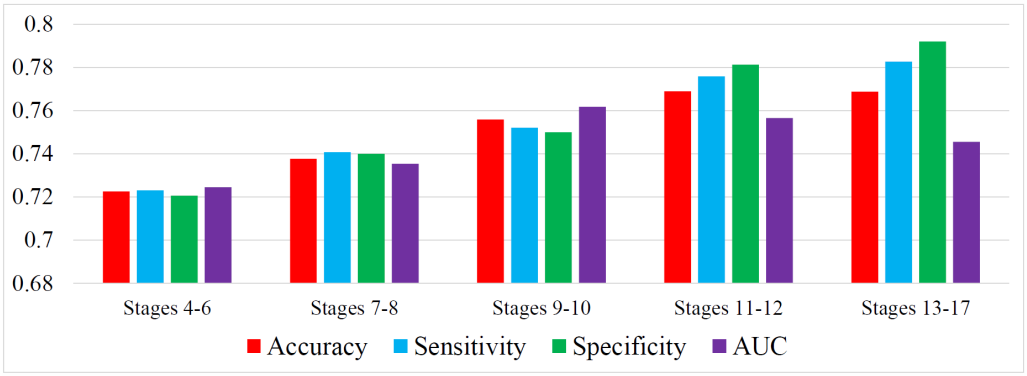


(c) ODL ($m = 2000$)

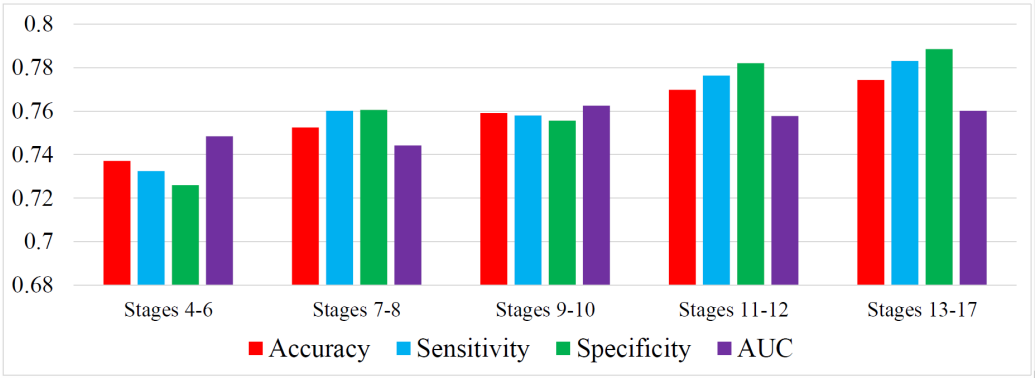
Figure 5.3: Classification Performance of ODL



(a) SCC ($m = 500$)



(b) SCC ($m = 1000$)



(c) SCC ($m = 2000$)

Figure 5.4: Classification Performance of SCC

Chapter 6

MULTI-SOURCE MULTI-TARGET DICTIONARY LEARNING FOR PREDICTION OF COGNITIVE DECLINE

6.1 Introduction

In this chapter, I propose a multi-task dictionary learning method to predict the cognitive decline of AD patients. Recently, Multi-task sparse feature learning has been successfully applied to many computer vision and biomedical informatics researches. It aims to improve the generalization performance by exploiting the shared features among different tasks. However, most of the existing algorithms are formulated as a supervised learning scheme. Its drawback is with either insufficient feature numbers or missing label information. To address these challenges, I formulate an unsupervised framework for multi-task sparse feature learning based on a novel dictionary learning algorithm. To solve the unsupervised learning problem, I propose a two-stage Multi-Source Multi-Target Dictionary Learning (MMDL) algorithm. In stage 1, I propose a multi-source dictionary learning method to utilize the common and individual sparse features in different time slots. In stage 2, supported by a rigorous theoretical analysis, I develop a multi-task learning method to solve the missing label problem.

The slow progressive neurodegenerative disorder of AD leads to a loss of memory and reduction of cognitive function. Many clinical/cognitive measures such as Mini Mental State Examination (MMSE) and Alzheimer's Disease Assessment Scale cognitive subscale (ADAS-Cog) have been designed to evaluate a subject's cognitive decline. Subjects are commonly divided into three different groups: AD, Mild Cognitive Impairment (MCI) and Cognitively Unimpaired (CU), defined clinically based on

behavioral and above assessments. It is crucial to predict AD related cognitive decline so an early intervention or prevention becomes possible. Prior research have shown measures from brain magnetic resonance (MR) images correlate closely with cognitive changes and have great potentials to provide early diagnostic markers to predict cognitive decline presymptomatically in a sufficiently rapid and rigorous manner.

The main challenge in AD diagnosis or prognosis with neuroimaging arises from the fact that the data dimensionality is intrinsically high while only a small number of samples are available. In this regard, machine learning has been playing a pivotal role to overcome this so-called "large p, small n" problem. A dictionary that allows us to represent original features as superposition of a small number of its elements so that I can reduce high dimensional image to a small number of features. Dictionary learning Lee *et al.* (2006) has been proposed to use a small number of basis vectors to represent local features effectively and concisely and help image content analysis. However, most existing works on dictionary learning focused on the prediction of target at a single time point Zhang *et al.* (2016b); Nie *et al.* (2015) or some region-of-interest Zhang *et al.* (2016a,c, 2017a). In general, a joint analysis of tasks from multiple sources is expected to improve the performance but remains challenging.

Although a general unsupervised dictionary learning may overcome the missing label problem to obtain the sparse features, I still need to consider the prediction labels at different time points after I learn the sparse features. A forthright method is to perform linear regression at each time point and determine weighted matrix W separately. However, even when I have the common dictionary which models the relationship among different tasks, if prediction is purely based on linear regression which treats all tasks independently and ignores the useful information reserved in the change along the time continuum, there still exists strong bias to predict future multiple targets clinical scores.

To excavate the correlations among the cognitive scores, several multi-task models were put forward. Wang *et al.* (2011a) proposed a sparse multi-task regression and feature selection method to jointly analyze the neuroimaging and clinical data in prediction of the memory performance. Zhang and Shen Zhang *et al.* (2012) exploited a $l_{2,1}$-norm based group sparse regression method to select features that could be used to jointly represent the different clinical status and two clinical scores (MMSE and ADAS-cog). Xiang *et al.* (2014) proposed a sparse regression-based feature selection method for AD/MCI diagnosis to maximally utilize features from multiple sources by focusing on a missing modality problem. However, the clinical scores for many patients are missing at some time points, i.e., the target vector y_i may be incomplete and the above methods all failed to model this issue. A simple strategy is to remove all patients with missing target values. It, however, significantly reduces the number of samples. Zhou *et al.* (2012) considered multi-task with missing target values in the training process, but the algorithm did not incorporate multiple sources data.

In this study, I propose a novel integrated unsupervised framework, termed Multi-Source Multi-Target Dictionary Learning (MMDL) algorithm, I utilize shared and individual dictionaries to encode both consistent and changing imaging features along longitudinal time points. Meanwhile, I also formulate different time point clinical score predictions as multi-task learning and overcome the missing target values in the training process. The pipeline of our method is illustrated in Fig 6.1. I evaluate the proposed framework on the $N = 3970$ longitudinal images from Alzheimer's Disease Neuroimaging Initiative (ADNI) database and use longitudinal hippocampal surface features to predict future cognitive scores. Our experimental results outperform some other state-of-the-art methods and demonstrate the effectiveness of the proposed algorithm.

Our main contributions can be summarized into threefold. Firstly, I considered

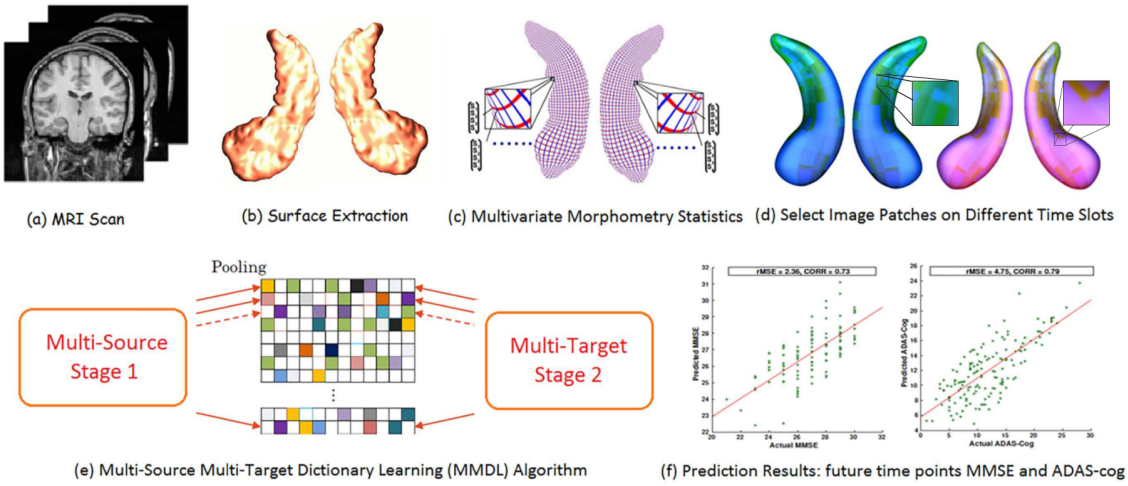


Figure 6.1: The Pipeline of Our Method. I Extracted Hippocampi From MRI Scans (a), Then I Registered Hippocampal Surfaces (b) and Computed Surface Multivariate Morphometry Statistics (c). Image Patches Were Extracted From the Surface Maps to Initialize the Dictionary (d) for Multi-Source Multi-Target Dictionary Learning (e). I Used Features from Two Time Points to Predict Five Future Time Points MMSE and ADAS-cog (f).

the variance of subjects from different time points (Multi-Source) and proposed an unsupervised dictionary learning method in stage 1 of the MMDL algorithm, in which not only does a patient share features between different time slots but different patients share some common features within the same time point. I also explore the relationship between the shared and individual dictionary in stage 1. Secondly, I use sparse features learned from dictionary learning as an input and multiple future clinical scores as corresponding labels (Multi-Target) to train the multi-task prediction model in stage 2 of the MMDL Algorithm. To the best of our knowledge, it is the first learning model which unifies both multiple source inputs and multiple target outputs with dictionary learning research for brain imaging analysis. Lastly, I also take into account the incomplete label problem. I deal with the missing label

problem during the regression process and theoretically prove the correctness of the regression model. Our extensive experimental results on the ADNI dataset show the proposed MMDL achieves faster running speed and lower estimation errors, as well as reasonable prediction scores when comparing with other state-of-the-art algorithms.

6.2 Multi-Source Multi-Target Dictionary Learning

Stage 1: Multi-Source Dictionary Learning Stage

Given subjects from T time points: $\{X_1, X_2, ..., X_T\}$, our goal is to learn a set of sparse codes $\{Z_1, Z_2, ..., Z_T\}$ for each time point where $X_t \in \mathbb{R}^{p \times n_t}$, $Z_t \in \mathbb{R}^{l_t \times n_t}$ and $t \in \{1, ..., T\}$. p is the feature dimension of each subject, n_t is the number of subjects for X_t and l_t is the dimension of each sparse code in Z_t. When employing the online dictionary learning (ODL) method Mairal *et al.* (2009) to learn the sparse codes Z_t by X_t individually, I obtain a set of dictionary $\{D_1, ..., D_T\}$ but there is no correlation between learnt dictionaries. Another solution is to construct the subjects $\{X_1, ..., X_T\}$ into one data matrix X to obtain the dictionary D. However, only one dictionary D is not sufficient to model the variations among subjects from different time points. To address this problem, I integrate the idea of multi-task learning into the ODL method. I propose a novel online dictionary learning algorithm, called Multi-Source Multi-Target Dictionary Learning (MMDL), to learn the subjects from different time points.

For the subject matrix X_t of a particular time point, MMDL learns a dictionary D_t and sparse codes Z_t. D_t is composed of two parts: $D_t = [\hat{D}_t, \bar{D}_t]$ where $\hat{D}_t \in \mathbb{R}^{p \times \hat{l}}$, $\bar{D}_t \in \mathbb{R}^{p \times \bar{l}_t}$ and $\hat{l} + \bar{l}_t = l_t$. $\hat{D}_t$ is the common dictionary among all the learnt dictionaries $\{D_1, ..., D_T\}$ while $\bar{D}_t$ is different from each other and only learnt from the corresponding matrix X_t. Therefore, objective function of MMDL can be

reformulated as follows:

$$\min_{\substack{D_1,\cdots,D_T\in\Psi_t \\ Z_1,\cdots,Z_T}} \sum_{t=1}^{T} \frac{1}{2}||X_t - [\hat{D}_t, \bar{D}_t]Z_t||_F^2 + \lambda \sum_{t=1}^{T} ||Z_t||_1, \text{ subjects to: } \hat{D}_1 = \cdots = \hat{D}_T \quad (6.1)$$

where $\Psi_t = \{D_t \in \mathbb{R}^{p\times l_t} : \forall j \in 1, ..., l_t, |||[D_t]_j||_2 \leq 1\}$ $(t = 1, 2, \cdots, T)$ and $[D_t]_j$ is the jth column of D_t.

Fig 6.2 illustrates the framework of MMDL with subjects of ADNI from three different time points which represents as X_1, X_2 and X_3, respectively. Through the multi-source dictionary learning stage of MMDL, I obtain the dictionary and sparse codes for subjects from each time point t: D_t and Z_t. In Stage 1, a dictionary D_t is composed by a shared part $\hat{D}_t$ and an individual part $\bar{D}_t$. In this example, $\hat{D}_1$, $\hat{D}_2$ and $\hat{D}_3$ are the same. For the individual part of dictionaries, MMDL learns different $\bar{D}_t$ only from the corresponding matrix X_t. I vary the number of columns $\bar{l}_t$ in $\bar{D}_t$ to introduce the variant in the learnt sparse codes Z_t. As a result, the feature dimensions of learnt sparse codes matrix Z_t are different from each other. Then I employ the max-pooling Boureau *et al.* (2010) method to extract the features and use extracted features to perform the regression across different time points.

The initialization of dictionaries in MMDL is critical to the whole learning process. I propose a random patch method to initialize the dictionaries from different time points. The main idea of the random patch method is to randomly select l image patches from n subjects $\{x_1, x_2, ..., x_n\}$ to construct D where $D \in \mathbb{R}^{p\times l}$. It is a similar way to perform the random patch approach in MMDL. In MMDL, the way I initialize $\hat{D}_t$ is to randomly select $\hat{l}$ subjects from subjects across different time points $\{X_1, \cdots, X_T\}$ to construct it. For the individual part of each dictionary, I randomly select $\bar{l}$ subjects from the corresponding matrix X_t to construct $\bar{D}_t$. After initializing dictionary D_t for each time point, I set all the sparse codes Z_t to be zero at the beginning. For each sample X_t at t-th time point, $X_t \in \mathbb{R}^{p\times n_t}$.

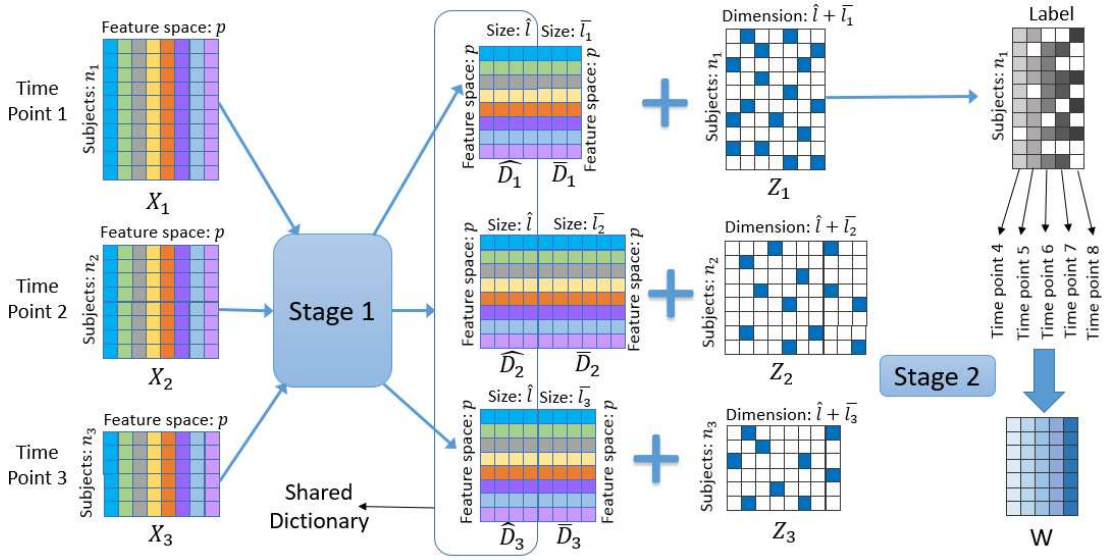


Figure 6.2: Illustration of the Learning Process of MMDL on ADNI Datasets From Multiple Different Time Points to Predict Multiple Future Time Points Clinical Scores.

Stage 2: Multi-Target Learning with Missing Label

In the longitudinal AD study, I measure the cognitive scores of selected patients at multiple time points. Instead of considering the prediction of cognitive scores at a single time point as a regression task, I formulate the prediction of clinical scores at multiple future time points as a multi-task regression problem. I employ multi-task regression formulations in place of solving a set of independent regression problems since the intrinsic temporal smoothness information among different tasks can be incorporated into the model as prior knowledge. However, the clinical scores for many patients are missing at some time points, especially for 36 and 48 months ADNI data. It is necessary to formulate a multi-task regression problem with missing target values to predict clinical scores.

In this study, I use a matrix $\Theta \in \mathbb{R}^{m_t \times n_t}$ to indicate missing target values, where

$\Theta_{i,j} = 0$ if the target value of label $Y_t(i,j)$ is missing and $\Theta_{i,j} = 1$ otherwise. Give the sparse codes $\{Z_1, ..., Z_T\}$ and corresponding labels $\{Y_1, ..., Y_T\}$ from different times where $Y_t \in \mathbb{R}^{m_t \times n_t}$, I formulate the multi-target learning stage with missing target values as:

$$\min_{W_1,\cdots,W_T} \sum_{t=1}^{T} ||\Theta(Y_t - W_t Z_t)||_F^2 + \xi \sum_{t=1}^{T} ||W_t||_F^2 \tag{6.2}$$

Although the Eqn. 6.2 is associated with missing values on the labels, I show that it has a close form and present the theoretical analysis of stage 2 as follows:

Theorem *For the data matrix pair (Z_t, Y_t), I denote the jth row's labels $\widetilde{Y}_t(j)$ in Y_t. I use $\widetilde{Z}_t$ and $\widetilde{Y}_t(j)$ to represent the remaining datasets after removing the missing value in $Y_t(j)$. The problem of (Eqn. 6.2) can decomposed as the following equation:*

$$\min_{w_t(j)} ||(\widetilde{Y}_t(j) - w_t(j)\widetilde{Z}_t)||_2^2 + \xi ||w_t(j)||_2^2 \tag{6.3}$$

Proof Eqn (6.3) is known the Ridge regression Hoerl and Kennard (1970). To optimize the problem, I calculate the gradient and set the gradient to be zero. Then I can get the optimal $w_t(j)$ by the following steps:

$$2\widetilde{Z}_t(\widetilde{Z}_t^T w_t(j) - \widetilde{Y}_t(j)) + 2\xi w_t(j) = 0$$

$$\widetilde{Z}_t\widetilde{Z}_t^T w_t(j) - \widetilde{Z}_t\widetilde{Y}_t(j) + \xi w_t(j) = 0$$

$$(\widetilde{Z}_t\widetilde{Z}_t^T + \xi I) w_t(j) = \widetilde{Z}_t\widetilde{Y}_t(j)$$

$$w_t(j) = (\widetilde{Z}_t\widetilde{Z}_t^T + \xi I)^{-1}\widetilde{Z}_t\widetilde{Y}_t(j) \qquad \square$$

After solving $w_t(j)$ for every time point where $j \in \{1, ..., m_t\}$, I can obtain the learnt model $\{W_1, ..., W_T\}$ to predict the clinical scores.

Our MMDL algorithm can be summarized into Algorithm 8. k denotes the epoch number where $k \in \{1, ..., \kappa\}$. Φ represents the shared part of each dictionary D_t which is initialized by the random patch method. For each image patch $x_t(i)$ extracted from

Algorithm 8 Multi-Source Multi-Target Dictionary Learning (MMDL)

Require: Samples and corresponding labels from different time points: $\{X_1, X_2, ...X_T\}$ and $\{Y_1, Y_2, ...Y_T\}$

Ensure: The model for different time points: $\{W_1, ..., W_T\}$.

Stage 1: Multi-Source Dictionary Learning

for $k = 1$ to κ **do**

  For each image patch $x_t(i)$ from sample X_t, $i \in \{1, ..., n_t\}$ and $t \in \{1, ..., T\}$.

  Update $\hat{D}_t^k$: $\hat{D}_t^k = \Phi$.

  Update $z_t^{k+1}(i)$ and index set $I_t^{k+1}(i)$ by a few steps of CCD:

    $[z_t^{k+1}(i), I_t^{k+1}(i)] = CCD(\hat{D}_t^k, \bar{D}_t^k, x_t(i), I_t^k(i), z_t^k(i))$.

  Update the $\hat{D}_t$ and $\bar{D}_t$ by one step SGD:

    $[\hat{D}_t^{k+1}, \bar{D}_t^{k+1}] = SGD(\hat{D}_t^k, \bar{D}_t^k, x_t(i), I_t^{k+1}(i), z_t^{k+1}(i))$.

  Normalize $\hat{D}_t^{k+1}$ and $\bar{D}_t^{k+1}$ based on the index set $I_t^{k+1}(i)$.

  Update the shared dictionary Φ: $\Phi = \hat{D}_t^{k+1}$.

end for

Obtain the learnt dictionaries and sparse codes: $\{D_1, ..., D_T\}$, $\{Z_1, ..., Z_T\}$.

Stage 2: Multi-Target Regression with incomplete label

for $t = 1$ to T **do**

  Given the jth column $Y_t(j)$ in Y_t, for the jth model $w_t(j)$ in W_t

  $w_t(j) = (\widetilde{Z}_t \widetilde{Z}_t^T + \xi I)^{-1} \widetilde{Z}_t \widetilde{Y}_t(j)$

end for

X_t, I learn the i-th sparse code $z_t^{k+1}(i)$ from Z_t by several steps of Cyclic Coordinate Descent (CCD) Canutescu and Dunbrack (2003). Then I use learnt sparse codes $z_t^{k+1}(i)$ to update the dictionary $\hat{D}_t^{k+1}$ and $\bar{D}_t^{k+1}$ by one step Stochastic Gradient Descent (SGD)Zhang (2004). Since $z_t^{k+1}(i)$ is very sparse, I use the index set $I_t^{k+1}(i)$ to record the location of non-zero entries in $z_t^{k+1}(i)$ to accelerate the update of sparse codes and dictionaries. Φ is updated by the end of the k-th iteration to ensure $\hat{D}_t^{k+1}$ is the same part among all the dictionaries.

Updating the Sparse Codes

After I pick an image patch $x_t(i)$ from the sample X_t at the time point t, I fix the dictionary and update the sparse codes by following the ODL method. Then the optimization problem I need to solve becomes the following equation:

$$\min_{z_t(i)} F(z_t(i)) = \frac{1}{2}||x_t(i) - [\hat{D}_t, \bar{D}_t]z_t(i)||_2^2 + \lambda||z_t(i)||_1. \tag{6.4}$$

It is known as the Lasso problem Tibshirani (1996). Coordinate descent Canutescu and Dunbrack (2003) is known as one of the state-of-the-art methods for solving this problem. In this study, I perform the CCD to optimize Eqn (6.4). Empirically, the iteration may take thousands of steps to converge, which is time-consuming in the optimization process of dictionary learning. However, I observed that after a few steps, the support of the coordinates, i.e., the locations of the non-zero entries in $z_t(i)$, becomes very accurate, usually after less than ten steps. In this study, I perform P steps CCD to generate the non-zero index set I_t^{k+1}, recording the non-zero entry of $z_t^{k+1}(i)$. Then I perform S steps CCD to update the sparse codes only on the non-zero entries of $z_t^{k+1}(i)$, accelerating the learning process significantly. SCC Lin *et al.* (2014); Lv *et al.* (2017) employs a similar strategy to update the sparse codes in a single task. For the multi-task learning, I summarize the updating rules as follows:

(a) Perform P steps CCD to update the locations of the non-zero entries $I_t^{k+1}(i)$ and the model $z_t(i)^{k+1}$.

(b) Perform S steps CCD to update the $z_t(i)^{k+1}$ in the index of $I_t^{k+1}(i)$.

In (a), for each step CCD, I will pick up j-th coordinate to update the model $z_t(i)_j$ and non-zero entries, where $j \in \{1, ..., l_t\}$. I perform the update from the 1st coordinate to the l_t-th coordinate. For each coordinate, I calculate the gradient g based on the objective function (6.4) then update the model $z_t^{k+1}(i)_j$ based on g. The calculation of g and $z_t^{k+1}(i)_j$ follows the equations:

$$g = [\hat{D}_t^k, \bar{D}_t^k]_j^T(\Omega([\hat{D}_t^k, \bar{D}_t^k], z_t^k(i), I_t^k(i)) - x_t(i)), \tag{6.5}$$

$$z_t^{k+1}(i)_j = \Gamma_\lambda(z_t^k(i)_j - g), \tag{6.6}$$

where Ω is a sparse matrix multiplication function that has three input parameters. Take $\Omega(A, b, I)$ as an example, A denotes a matrix, b is a vector and I is an index set that records the locations of non-zero entries in b. The returning value of function Ω is defined as: $\Omega(A, b, I) = Ab$. When multiplying A and b, I only manipulate the non-zero entries of b and corresponding columns of A based on the index set I, speeding up the calculation by utilizing the sparsity of b. Γ is the soft thresholding shrinkage function Combettes and Wajs (2005) and the definition of Γ is given by:

$$\Gamma_\varphi(x) = sign(x)(|x| - \varphi). \tag{6.7}$$

In the end of (a), I count the non-zero entries in $z_t^{k+1}(i)$ and store the non-zero index in $I_t^{k+1}(i)$. In (b), I perform S steps CCD by only considering the non-zero entries in $z_t^{k+1}(i)$. As a result, for each index μ in $I_t^{k+1}(i)$, I calculate the gradient g and update the $z_t^{k+1}(i)_\mu$ by:

$$g = [\hat{D}_t^k, \bar{D}_t^k]_\mu^T(\Omega([\hat{D}_t^k, \bar{D}_t^k], z_t^{k+1}(i), I_t^{k+1}(i)) - x_t(i)), \tag{6.8}$$

$$z_t^{k+1}(i)_\mu = \Gamma_\lambda((z_t^{k+1}(i)_\mu - g). \tag{6.9}$$

Since I only focus on the non-zero entries of the model and P is less than 10 iteration and S is a much larger number, I accelerate the learning process of sparse codes significantly.

Updating the Dictionaries

I update the dictionaries by fixing the sparse codes and updating the current dictionaries. Then, the optimization problem becomes as follow:

$$\min_{\hat{D}_t, \bar{D}_t} F(\hat{D}_t, \bar{D}_t) = \frac{1}{2}||x_t(i) - [\hat{D}_t, \bar{D}_t]z_t(i)||_2^2 \tag{6.10}$$

After I update the sparse codes, I have already known the non-zero entries of $z_t^{k+1}(i)$. Another key insight of MMDL is that I just need to focus on updating the non-zero entries of the dictionaries but not all columns of the dictionaries, and it accelerates the optimization dramatically. For example, when I update the i-th column and j-th row's entry of the dictionary D, the gradient of $D_{j,i}$ is set to be $\nabla D_{j,i} = z_i(D_j^T z - x_j)$. If the i-th entry of z is equal to zero, the gradient would be zero. As a result, I do not need to update the i-th column of the dictionary D. The learning rate is set to be an approximation of the inverse of the Hessian matrix H_t^{k+1}, which is updated by the sparse codes $z_t^{k+1}(i)$ in k-th iteration. In the beginning, I update the Hessian matrix by:

$$H_t^{k+1} = H_t^k + z_t^{k+1}(i)z_t^{k+1}(i)^T. \tag{6.11}$$

I perform one step SGD to update the dictionaries: $\hat{D}_t^{k+1}$ and $\bar{D}_t^{k+1}$. To speed up the computation, I use a vector to store the information $Dz - x$:

$$R = \Omega([\hat{D}_t^k, \bar{D}_t^k], z_t^{k+1}(i), I_t^{k+1}(i)) - x_t(i). \tag{6.12}$$

For entry of dictionary in the μ-th column and j-th row, the procedure of learning dictionaries take the form of

$$[\hat{D}_t^{k+1}, \bar{D}_t^{k+1}]_{j,\mu} = [\hat{D}_t^k, \bar{D}_t^k]_{j,\mu} - \frac{1}{H_t^{k+1}(\mu,\mu)} z_t^{k+1}(i)_\mu R_j, \tag{6.13}$$

where μ is the non-zero entry stored in $I_t^{k+1}(i)$. For the μ-th column of dictionary, I set the learning rate as the inverse of the diagonal element of the Hessian matrix, which is $1/H_t^{k+1}(\mu,\mu)$

Due to $D_t \in \Psi_t$ in equation (6.1), it is necessary to normalize the dictionaries $\hat{D}_t^{k+1}$ and $\bar{D}_t^{k+1}$ after updating them. I can perform the normalization on the corresponding columns of non-zero entries from $z_t^{k+1}(i)$ because the dictionaries updating only occurs on these columns. Utilizing the non-zero information from $I_t^{k+1}(i)$ can accelerate the whole learning process significantly.

6.3 Experiments

Experimental Setting

I studied multiple time points structural MR Imaging from ADNI baseline (837) and 6-month (733) datasets. The responses are the MMSE and ADAS-cog coming from 5 different time points: M12, M18, M24, M36 and M48. Thus, I learned a total of 3970 images which combines 2 sources and 5 targets. The sample sizes corresponding to 5 targets are 728, 326, 641, 454 and 251. For the experiments, I used hippocampal surface multivariate statistics Wang *et al.* (2011b) as learning features, which is a 4×1 vector on each vertex of 15000 vertices on every hippocampal surface.

I built a prediction model for the above datasets using MMDL algorithm. The code of MMDL is available on the GitHub [1] . To train the prediction models, 1102 patches of size 10×10 are extracted from surface mesh structures and each patch

[1]https://github.com/liohzhee/Multi-Task-Dictionary-Learning

dimension is 400. The model was trained on an Intel(R) Core(TM) i7-6700 K CPU with 4.0GHz processors, 64 GB of globally addressable memory and a single Nvidia GeForce GTX TITAN X GPU. In the experimental setting of Stage 1 in MMDL, the sparsity $\lambda = 0.1$. Also, I selected 10 epochs with a batch size of 1 and 3 iterations of CCD (P is set to be 1 and S is 3). When the dictionaries and sparse codes were learned, Max-Pooling was used to generate features for annotation and get a 1×1000 vector feature for each images. In the Stage 2, 5-fold cross validation is used to select model parameters ξ in the training data (between 10^{-3} and 10^{3}).

In order to evaluate the model, I randomly split the data into training and testing sets using a 9:1 ratio and used 10-fold cross validation to avoid data bias. Lastly, I evaluated the overall regression performance using weighted correlation coefficient (wR) and root mean square error (rMSE) for task-specific regression performance measures. The two measures are defined as $wR(Y, \hat{Y}) = \sum_{i=1}^{t} Corr(Y_i, \hat{Y}_i) n_i / \sum_{i=1}^{t} n_i$, $rMSE(y, \hat{y}) = \sqrt{||y - \hat{y}||_2^2 / n}$. For wR, Y_i is the ground truth of target of task i and $\hat{Y}_i$ is the corresponding predicted value, $Corr$ is the correlation coefficient between two vectors and n_i is the number of subjects of task i. For each task of rMSE, y and n is the ground truth of target and the number of subjects and $\hat{y}$ is the corresponding prediction. The smaller rMSE, the bigger wR mean the better results. I report the mean and standard deviation based on 50 iterations of experiments on different splits of data.

I compared MMDL with multiple state-of-the-art methods, ODL-L: the single-task online dictionary learning Mairal *et al.* (2009) followed by Lasso, L21: the multi-task method called $L_{2,1}$ norm regularization with least square loss Argyriou *et al.* (2008). TGL: the disease multi-task progression model called Temporal group Lasso Zhou *et al.* (2012), as well as Ridge and Lasso. For the parameters selection, I used the same method with the experimental setting in our stage 2.

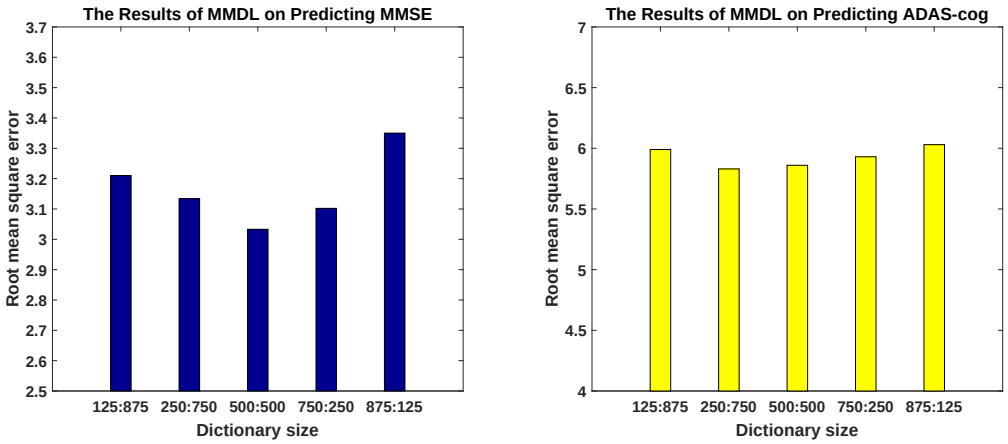


Figure 6.3: Comparison of rMSE Performance by Varying the Size of Common Dictionary.

Experimental Results

The Size of Common Dictionaries in MMDL. In Stage 1 of MMDL, the common dictionary is assumed to be shared by different tasks. It is necessary to evaluate what is an appropriate size of such common dictionary. Therefore, I set the dictionary size to be 1000 and partitioned the dictionary by different proportions: 125:875, 250:750,500:500, 750:250 and 875:125, where the left number is the size of common dictionary while the right one is the size of individual dictionary for each task. Fig 6.3 shows the results of rMSE of MMSE and ADAS-cog prediction. As it shows in Fig 6.3, the rMSE of MMSE and ADAS-Cog are lowest when I split the dictionary by half and a half. It means the both of common and individual dictionaries are of equal importance during the multi-task learning. In all experiments, I use the split of 500:500 as the size of common and individual dictionaries, the dimension of each sparse code in MMDL is 1000.

Time Efficiency Comparison I compare the efficiency of our proposed MMDL with the state-of-the-art online dictionary learning (ODL). In this experiment, I focus

Table 6.1: Time Comparisons of MMDL and ODL by Varying Dictionary Size.

Dictionary Size	MMDL	ODL
500	1.74 hour	8.84 hour
1000	3.34 hour	21.95 hour
2000	6.93 hour	49.90 hour

on the single batch size setting, that is, I process one image patch in each iteration. I vary the dictionary size as: 500, 1000 and 2000. For MMDL, the ratio between the common dictionary and the individual parts is 1:1. I report the results in Table 6.1. I observe that the proposed MMDL use less time than ODL. When the size of dictionary are increasing, MMDL is more efficient and has a higher speedup compared to ODL.

Performance Comparison I report the results of MMDL and other methods on the prediction model of MMSE with ADNI group in Table 6.2. The proposed approach MMDL outperformed ODL-L, Lasso and Ridge, in terms of both rMSE and correlation coefficient wR on four different time points. The results of Lasso and Ridge are very close while sparse coding methods are superior to them. For sparse coding models, I observe that MMDL obtained a lower rMSE and higher correlation result than traditional sparse coding method ODL-L since I consider the correlation between different time slots for different tasks and the relationship with different time points on the same patient among all tasks. I also notice that the proposed MMDL's significant accuracy improvement for later time points. This may be due to the data sparseness in later time points, as the proposed sparsity-inducing models are expected to achieve better prediction performance in this case.

I follow the same experimental procedure in the MMSE study and explore the

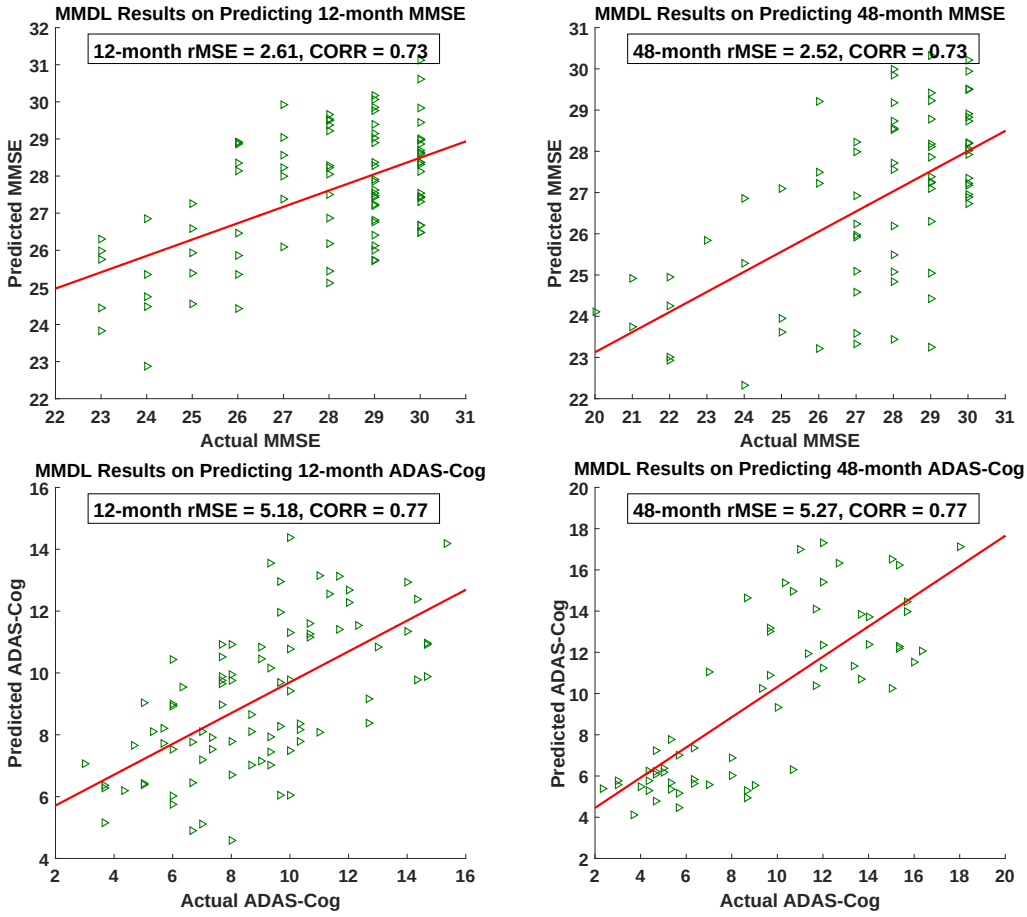


Figure 6.4: Scatter Plots of Actual MMSE and ADAS-Cog Versus Predicted Values on M12 and M48 by Using MMDL.

prediction model by ADAS-cog scores. The prediction performance results are shown in Table 6.3. I can observe that the best performance of predicting scores of ADAS-Cog is achieved by MMDL for four time points.

Comparing with L21, after MMDL dealing with missing label, the results more linear, reasonable and accurate. Due to the dimension of M36 and M48 is too small, it is hard to learn a complete model. TGL also considered the issue of missing labels, however, MMDL still achieved the better results because MMDL incorporates multiple sources data and uses common and individual dictionaries. Although the result

of MMDL had bias, MMDL still achieved the best result compared with the other five methods on predicting both MMSE and ADAS-cog, which shows our method is more efficient about dealing with missing data.

I show the scatter plots for the predicted values versus the actual values for MMSE and ADAS-Cog on the M12 and M48 in Fig 6.4. In the scatter plots, I see the predicted values and actual clinical scores have a high correlation. The scatter plots show that the prediction performance for ADAS-Cog is better than that of MMSE.

6.4 Summary

In this study, I propose a novel Multi-Source Multi-Target Dictionary Learning for modeling cognitive decline, which allows simultaneous selections of a common set of biomarkers for multiple time points and specific sets of biomarkers for different time points using dictionary learning. I consider predicting future clinical scores as multi-task and deal with the missing labels problem. The effectiveness of the proposed progression model is supported by extensive experimental studies. The experimental results demonstrate that the proposed progression model is more effective than other state-of-the-art methods.

Table 6.2: The Prediction Results of MMSE on Whole Dataset.

Methods	wR	M12	M18	M24	M36	M48
Lasso	0.40±0.09	4.04±0.77	3.46±0.97	5.53±0.86	4.39±0.74	4.73±1.49
Ridge	0.41±0.07	4.26±0.56	3.56±0.93	5.05±0.54	4.21±0.47	3.62±0.91
L21	0.57±0.01	3.32±0.63	4.75±0.75	4.64±0.88	4.08±1.01	3.11±1.05
ODL-L	0.63±0.08	2.99±0.63	**2.88±0.68**	4.29±0.84	3.62±1.45	2.93±1.07
TGL	0.70±0.05	2.73±0.72	4.00±1.31	4.00±0.64	3.19±1.38	2.60±1.42
MMDL	**0.73±0.02**	**2.61±0.55**	3.37±1.01	**3.66±0.78**	**2.73±1.09**	**2.52±1.20**

Table 6.3: The Prediction Results of ADAS-cog on Whole Dataset.

Methods	wR	M12	M18	M24	M36	M48
Lasso	0.49±0.05	6.81±1.03	6.87±0.74	7.62±0.87	8.08±1.39	6.55±1.34
Ridge	0.46±0.07	7.68±0.96	6.89±1.69	7.84±1.54	8.59±0.62	6.64±1.58
L21	0.53±0.07	6.40±0.51	6.95±0.88	8.07±0.67	8.00±1.04	5.92±0.60
ODL-L	0.53±0.05	5.65±0.73	4.97±0.67	7.30±0.77	7.25±0.69	5.56±1.22
TGL	0.72±0.04	5.52±1.15	5.70±0.53	6.85±1.06	**6.36±1.22**	5.73±0.61
MMDL	**0.77±0.02**	**5.18±0.88**	**4.64±1.12**	**6.76±1.35**	6.78±1.54	**5.27±1.76**

Chapter 7

CONCLUSION

In this chapter, I summarize the major contributions of this dissertation. Moreover, I will discuss some of the possible future research directions.

7.1 Summary

Sparse learning is a widely used technique in data mining and machine learning for model selection and feature extraction. Regularization has been commonly employed to obtain more stable and interpretable models. For the regularized sparse models, the ℓ_1 norm regularization has achieved great success in many applications, such as Lasso regression Tibshirani (1996). The issue to apply the sparse learning on real world application is limited by the huge dimension in the sample and feature space. The major theme of this dissertation is to scale up the optimization of sparse learning in terms of computational resource, such as multiple-threaded computing and distributed computing, or utilizing the sparse characteristic of learnt models to accelerate the optimization.

Firstly, I propose a parallel framework to scale up the ℓ_1-norm regularized loss minimization problem in a multithreading environment. In this framework, I propose an asynchronous solver (AGCD) to reduce the waiting time when multiple threads update the model concurrently. The proposed AGCD adopts a grouped selection strategy to update the coordinates rather than random selection. The selected coordinate has to win the competition among a group of candidates to get updated, resolving the situation that parallel solvers might diverge with a small feature space. Moreover, I propose the parallel screening methods to pre-identify the remove the

inactive features from the optimization, which would result in substantial savings in computational cost.

Secondly, I propose a distributed framework to scale up the sparse regression model with multiple machines. Since the data set are distributed among different institutions, I propose a Local Query Model (LQM) to properly maintain data privacy for all the institutions. Then the model is learnt by employing the Accelerated Gradient Method (AGM) in a distributed manner. I develop a family of distributed Lasso screening rule to reduce the feature space since the data set are distributed among different institutions. The proposed model is integrated with detecting the risk genetics factors of AD. In addition, I introduce a group feature selection framework to select the relevant group features for the imaging genetics studies of risk genetic factors by optimizing the distributed group Lasso in a sequence of regularized parameter values.

Finally, for many biological and medical imaging problems, the current bottleneck is the limited amount of available labeled training data. In this dissertation, I propose a novel unsupervised sparse learning solver (SCC), scaling up the optimization of dictionary learning and sparse coding problems. SCC demonstrates the effectiveness and efficiency for the computer vision and real world application, such as the Drosophila gene expression pattern annotation. Moreover, another issue for the medical imaging research is to deal with the longitudinal features of patients for different time points. I propose a multi-task dictionary learning method to learn the different time point tasks simultaneously by partitioning the dictionaries into the common and individual parts, considering the variance of subjects from multiple time points or multiple ROIs. The proposed method is evaluated by predicting the cognitive decline of AD patients, achieving lower MMSE and ADAS-cog scores compared to other state-of-the-art methods.

7.2 Future Directions

For further investigation, the following directions appear promising.

Many researches have shown that solving the dual problem using coordinate descent methods is faster on large-scale optimization problems. For example, Stochastic Dual Coordinate Ascent (SDCA) Shalev-Shwartz and Zhang (2013); Johnson and Zhang (2013); Shalev-Shwartz and Zhang (2014) has become the most widely-used algorithm for big data optimization. It is interesting to further investigate the idea to scale up advanced solvers such as SDCA on the optimization of sparse learning problems. There are already existing works like PASSCoDe Hsieh *et al.* (2015) and SA-SDCA Tran *et al.* (2015), parallelizing SDCA in a multithreading environment. Although these parallel solvers adopt the asynchronous method to scale up the optimization, the practical usage and scalability is restricted by high dimensionality in the feature space. This dissertation provides a way to resolve the curse-of-dimensionality via scaling up the optimization of sparse learning. I have shown such a possibility to scale up the SDCA in the same direction. Moreover, it is a challenge to scale up the coordinate descent methods in a distributed environment. Zhang and Kwok (2014) proposed an asynchronous distributed ADMM method for consensus optimization. It is essential to employ the asynchronous technology for the optimization of SDCA in a distributed manner. A good proposal is to choose multiple coordinates to update in each iteration then update the step size at the end of each iteration, guaranteeing the convergence of the distributed solver.

To scaling up the unsupervised sparse learning problems, such as the dictionary learning, I have proposed an efficient solver SCC to resolve the optimization problem. The future direction is to parallelize SCC in a multithreading or a distributed environment. The update of sparse codes is coordinate descent methods essentially. The

parallelization of coordinate descents is already resolved by synchronous and asynchronous parallel solvers Kyrola *et al.* (2011); Scherrer *et al.* (2012b); Liu *et al.* (2014); Hsieh *et al.* (2015); Li *et al.* (2016a, 2017). SCC adopts one step SGD to update the dictionary. The parallelization of one step SGD is straightforward. Therefore, this dissertation shows a possibility to parallelize the dictionary learning and sparse coding models in the same direction. The parallelization of multi-task sparse codings Zhang *et al.* (2017b) could follow the same way to achieve it. To scale up the dictionary in a distributed manner, Li *et al.* (2016c) proposed a novel distributed dictionary learning method to scale up the optimization of dictionary learning on Apache Spark. However, there are still some issues to be resolved such as reducing the data communication between the computation nodes. This dissertation provides a method to conduct the research along this direction.

REFERENCES

Argyriou, A., T. Evgeniou and M. Pontil, "Convex multi-task feature learning", Machine Learning **73**, 3, 243–272 (2008).

Association, A. *et al.*, "2011 alzheimer's disease facts and figures.", Alzheimer's & dementia: the journal of the Alzheimer's Association **7**, 2, 208 (2011).

Balakrishnan, S. and D. Madigan, "Algorithms for sparse linear classifiers in the massive data setting", Journal of Machine Learning Research **9**, Feb, 313–337 (2008).

Balasubramanian, K., K. Yu and G. Lebanon, "Smooth sparse coding via marginal regression for learning sparse representations.", in "Proceedings of the 30th International Conference on Machine Learning (ICML-13)", pp. 289–297 (2013).

Beck, A. and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems", SIAM journal on imaging sciences **2**, 1, 183–202 (2009).

Boureau, Y.-L., J. Ponce and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition", in "Proceedings of the 27th international conference on machine learning (ICML-10)", pp. 111–118 (2010).

Bousquet, O. and L. Bottou, "The tradeoffs of large scale learning", in "Advances in neural information processing systems", pp. 161–168 (2008).

Boyd, S., N. Parikh, E. Chu, B. Peleato and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers", Foundations and Trends® in Machine Learning **3**, 1, 1–122 (2011).

Burns, A., "Alzheimer's disease: on the verges of treatment and prevention", The Lancet Neurology **8**, 1, 4–5 (2009).

Canutescu, A. A. and R. L. Dunbrack, "Cyclic coordinate descent: A robotics algorithm for protein loop closure", Protein science **12**, 5, 963–972 (2003).

Chang, C.-C. and C.-J. Lin, "Libsvm: a library for support vector machines", ACM Transactions on Intelligent Systems and Technology (TIST) **2**, 3, 27 (2011).

Chen, J., L. Tang, J. Liu and J. Ye, "A convex formulation for learning shared structures from multiple tasks", in "Proceedings of the 26th Annual ICML", pp. 137–144 (ACM, 2009).

Chen, S. S., D. L. Donoho and M. A. Saunders, "Atomic decomposition by basis pursuit", SIAM review **43**, 1, 129–159 (2001).

Coates, A. and A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization", in "Proceedings of the 28th International Conference on Machine Learning (ICML-11)", pp. 921–928 (2011).

Combettes, P. L. and V. R. Wajs, "Signal recovery by proximal forward-backward splitting", Multiscale Modeling & Simulation **4**, 4, 1168–1200 (2005).

Corder, E. *et al.*, "Gene dose of apolipoprotein e type 4 allele and the risk of alzheimer's disease in late onset families", Science **261**, 5123, 921–923 (1993).

Daubechies, I., M. Defrise and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint", Communications on pure and applied mathematics **57**, 11, 1413–1457 (2004).

Dickerson, B. C., I. Goncharova, M. Sullivan, C. Forchetti, R. Wilson, D. Bennett, L. Beckett *et al.*, "Mri-derived entorhinal and hippocampal atrophy in incipient and very mild alzheimer's disease", Neurobiology of aging **22**, 5, 747–754 (2001).

Donoho, D. L. and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via 1 minimization", Proceedings of the National Academy of Sciences **100**, 5, 2197–2202 (2003).

Efron, B., T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression", The Annals of statistics **32**, 2, 407–499 (2004).

Evgeniou, T., C. A. Micchelli and M. Pontil, "Learning multiple tasks with kernel methods", in "Journal of Machine Learning Research", pp. 615–637 (2005).

Ghaoui, L. E., V. Viallon and T. Rabbani, "Safe feature elimination for the lasso and sparse supervised learning problems", Pacific Journal of Optimization , 8, 667–698 (2012).

Harold, D. *et al.*, "Genome-wide association study identifies variants at clu and picalm associated with alzheimer's disease", Nature genetics **41**, 10, 1088–1093 (2009).

Hoerl, A. E. and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems", Technometrics **12**, 1, 55–67 (1970).

Hsieh, C.-J., H.-F. Yu and I. S. Dhillon, "Passcode: Parallel asynchronous stochastic dual co-ordinate descent", in "Proceedings of the 31st International Conference on Machine Learning (ICML-15)", (2015).

Jarrett, K., K. Kavukcuoglu, Y. LeCun *et al.*, "What is the best multi-stage architecture for object recognition?", in "Computer Vision, 2009 IEEE 12th International Conference on", pp. 2146–2153 (IEEE, 2009).

Ji, S., L. Yuan, Y.-X. Li, Z.-H. Zhou, S. Kumar and J. Ye, "Drosophila gene expression pattern annotation using sparse features and term-term interactions", in "Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 407–416 (ACM, 2009).

Johnson, R. and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction", in "Advances in Neural Information Processing Systems", pp. 315–323 (2013).

Keerthi, S. S., K. Duan, S. K. Shevade and A. N. Poo, "A fast dual algorithm for kernel logistic regression", Machine learning **61**, 1-3, 151–165 (2005).

Kyrola, A., D. Bickson, C. Guestrin and J. K. Bradley, "Parallel coordinate descent for l_1-regularized loss minimization", in "Proceedings of the 28th International Conference on Machine Learning (ICML-11)", pp. 321–328 (2011).

LeCun, Y., L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE **86**, 11, 2278–2324 (1998).

Lee, H., A. Battle, R. Raina and A. Y. Ng, "Efficient sparse coding algorithms", in "Advances in neural information processing systems", pp. 801–808 (2006).

Lee, H., A. Battle, R. Raina and A. Y. Ng, "Efficient sparse coding algorithms", Advances in neural information processing systems **19**, 801 (2007).

Lewis, D. D., Y. Yang, T. G. Rose and F. Li, "Rcv1: A new benchmark collection for text categorization research", The Journal of Machine Learning Research **5**, 361–397 (2004).

Li, Q., S. Qiu, S. Ji, P. M. Thompson, J. Ye and J. Wang, "Parallel lasso screening for big data optimization", in "Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", pp. 1705–1714 (ACM, 2016a).

Li, Q., T. Yang, L. Zhan, D. P. Hibar, N. Jahanshad, Y. Wang, J. Ye, P. M. Thompson and J. Wang, "Large-scale collaborative imaging genetics studies of risk genetic factors for alzheimer's disease across multiple institutions", in "International Conference on Medical Image Computing and Computer-Assisted Intervention", pp. 335–343 (Springer, 2016b).

Li, Q., D. Zhu, J. Zhang, D. P. Hibar, N. Jahanshad, Y. Wang, J. Ye, P. M. Thompson and J. Wang, "Large-scale feature selection of risk genetic factors for alzheimer's disease via distributed group lasso regression", arXiv preprint arXiv:1704.08383 (2017).

Li, X., M. Makkie, B. Lin, M. Sedigh Fazli, I. Davidson, J. Ye, T. Liu and S. Quinn, "Scalable fast rank-1 dictionary learning for fmri big data analysis", in "Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", pp. 511–519 (ACM, 2016c).

Li, Y. and S. Osher, "Coordinate descent optimization for l^1 minimization with application to compressed sensing; a greedy algorithm", Inverse Problems and Imaging **3**, 3, 487–503 (2009).

Lin, B., Q. Li, Q. Sun, M.-J. Lai, I. Davidson, W. Fan and J. Ye, "Stochastic coordinate coding and its application for drosophila gene expression pattern annotation", arXiv preprint arXiv:1407.8147 (2014).

Liu, C.-C., T. Kanekiyo, H. Xu and G. Bu, "Apolipoprotein e and alzheimer disease: risk, mechanisms and therapy", Nature Reviews Neurology **9**, 2, 106–118 (2013).

Liu, J., S. Ji and J. Ye, *SLEP: Sparse Learning with Efficient Projections*, Arizona State University, URL `http://www.public.asu.edu/ jye02/Software/SLEP` (2009).

Liu, J., S. Wright, C. Re, V. Bittorf and S. Sridhar, "An asynchronous parallel stochastic coordinate descent algorithm", in "Proceedings of the 31st International Conference on Machine Learning (ICML-14)", pp. 469–477 (2014).

Lowe, D. G., "Object recognition from local scale-invariant features", in "Computer vision, 1999. The proceedings of the seventh IEEE international conference on", vol. 2, pp. 1150–1157 (Ieee, 1999).

Lv, J., B. Lin, Q. Li, W. Zhang, Y. Zhao, X. Jiang, L. Guo, J. Han, X. Hu, C. Guo *et al.*, "Task fmri data analysis based on supervised stochastic coordinate coding", Medical Image Analysis (2017).

Mace, D. L., N. Varnado, W. Zhang, E. Frise and U. Ohler, "Extraction and comparison of gene expression patterns from 2d rna in situ hybridization images", Bioinformatics **26**, 6, 761–769 (2010).

Mairal, J., F. Bach, J. Ponce and G. Sapiro, "Online dictionary learning for sparse coding", in "Proceedings of the 26th Annual International Conference on Machine Learning", pp. 689–696 (ACM, 2009).

Maurer, A., M. Pontil and B. Romera-Paredes, "Sparse coding for multitask and transfer learning", in "Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013", pp. 343–351 (2013).

Meinshausen, N. and P. Buhlmann, "Stability selection", Journal of the Royal Statistical Society: Series B (Statistical Methodology) **72**, 4, 417–473 (2010).

Mota, J. F., J. M. Xavier, P. M. Aguiar and M. Puschel, "Distributed basis pursuit", IEEE Transactions on Signal Processing **60**, 4, 1942–1956 (2012).

Nesterov, Y., "Efficiency of coordinate descent methods on huge-scale optimization problems", SIAM Journal on Optimization **22**, 2, 341–362 (2012).

Nie, Z., T. Yang, Y. Liu, B. Lin, Q. Li, V. A. Narayan, G. Wittenberg and J. Ye, "Melancholic depression prediction by identifying representative features in metabolic and microarray profiles with missing values", in "Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing", p. 455 (NIH Public Access, 2015).

Olshausen, B. A. and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images", Nature **381**, 6583, 607 (1996).

Olshausen, B. A. and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?", Vision research **37**, 23, 3311–3325 (1997).

Olshausen, B. A. and D. J. Field, "Sparse coding of sensory inputs", Current opinion in neurobiology **14**, 4, 481–487 (2004).

Peng, Z., M. Yan and W. Yin, "Parallel and distributed sparse optimization", in "Signals, Systems and Computers, Asilomar Conference on", pp. 659–646 (IEEE, 2013).

Qin, Z., K. Scheinberg and D. Goldfarb, "Efficient block-coordinate descent algorithms for the group lasso", Mathematical Programming Computation **5**, 2, 143–169 (2013).

Richtárik, P. and M. Takáč, "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function", Mathematical Programming **144**, 1-2, 1–38 (2014).

Richtárik, P. and M. Takáč, "Parallel coordinate descent methods for big data optimization", Mathematical Programming pp. 1–52 (2016).

Rouillard, A. D., G. W. Gundersen, N. F. Fernandez, Z. Wang, C. D. Monteiro, M. G. McDermott and A. Maayan, "The harmonizome: a collection of processed datasets gathered to serve and mine knowledge about genes and proteins", Database **2016**, baw100 (2016).

Sasieni, P. D., "From genotypes to genes: doubling the sample size", Biometrics pp. 1253–1261 (1997).

Scherer, D., A. Müller and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition", in "International Conference on Artificial Neural Networks", pp. 92–101 (Springer, 2010).

Scherrer, C., M. Halappanavar, A. Tewari and D. Haglin, "Scaling up coordinate descent algorithms for large regularization problems", in "Proceedings of the 29st International Conference on Machine Learning (ICML-12)", (2012a).

Scherrer, C., A. Tewari, M. Halappanavar and D. Haglin, "Feature clustering for accelerating parallel coordinate descent", in "Advances in Neural Information Processing Systems", pp. 28–36 (2012b).

Shalev-Shwartz, S. and A. Tewari, "Stochastic methods for l_1-regularized loss minimization", The Journal of Machine Learning Research **12**, 1865–1892 (2011).

Shalev-Shwartz, S. and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization", Journal of Machine Learning Research **14**, Feb, 567–599 (2013).

Shalev-Shwartz, S. and T. Zhang, "Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization.", in "Proceedings of the 31st International Conference on Machine Learning (ICML-14)", pp. 64–72 (2014).

Smith, E. C. and M. S. Lewicki, "Efficient auditory coding", Nature **439**, 7079, 978–982 (2006).

Szlam, A., K. Gregor and Y. LeCun, "Fast approximations to structured sparse coding and applications to object classification", in "European Conference on Computer Vision", pp. 200–213 (Springer, 2012).

Thompson, P. M. *et al.*, "The enigma consortium: large-scale collaborative analyses of neuroimaging and genetic data", Brain imaging and behavior **8**, 2, 153–182 (2014).

Tibshirani, R., "Regression shrinkage and selection via the lasso", Journal of the Royal Statistical Society. Series B (Methodological) pp. 267–288 (1996).

Tibshirani, R., J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor and R. J. Tibshirani, "Strong rules for discarding predictors in lasso-type problems", Journal of the Royal Statistical Society: Series B (Statistical Methodology) **74**, 2, 245–266 (2012a).

Tibshirani, R. *et al.*, "Strong rules for discarding predictors in lasso-type problems", Journal of the Royal Statistical Society: Series B **74**, 2, 245–266 (2012b).

Toga, A. W., K. L. Crawford, A. D. N. Initiative *et al.*, "The informatics core of the alzheimer's disease neuroimaging initiative", Alzheimer's & Dementia **6**, 3, 247–256 (2010).

Tran, K., S. Hosseini, L. Xiao, T. Finley and M. Bilenko, "Scaling up stochastic dual coordinate ascent", in "Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", pp. 1185–1194 (ACM, 2015).

Tsai, W.-T., C. J. Colbourn, J. Luo, G. Qi, Q. Li and X. Bai, "Test algebra for combinatorial testing", in "Automation of Software Test (AST), 2013 8th International Workshop on", pp. 19–25 (IEEE, 2013a).

Tsai, W.-T., Q. Li, C. J. Colbourn and X. Bai, "Adaptive fault detection for testing tenant applications in multi-tenancy saas systems", in "Cloud Engineering (IC2E), 2013 IEEE International Conference on", pp. 183–192 (IEEE, 2013b).

Wang, H., F. Nie, H. Huang, S. Risacher, C. Ding, A. J. Saykin, L. Shen *et al.*, "Sparse multi-task regression and feature selection to identify brain imaging predictors for memory performance", in "2011 International Conference on Computer Vision", pp. 557–562 (IEEE, 2011a).

Wang, J., Q. Li, S. Yang, W. Fan, P. Wonka and J. Ye, "A highly scalable parallel algorithm for isotropic total variation models", in "Proceedings of the 31st International Conference on Machine Learning (ICML-14)", pp. 235–243 (2014).

Wang, J. and J. Ye, "Two-layer feature reduction for sparse-group lasso via decomposition of convex sets", in "Advances in Neural Information Processing Systems", pp. 2132–2140 (2014).

Wang, J. and J. Ye, "Multi-layer feature reduction for tree structured group lasso via hierarchical projection", in "Advances in Neural Information Processing Systems", pp. 1279–1287 (2015a).

Wang, J. and J. Ye, "Safe screening for multi-task feature learning with multiple data matrices", in "Proceedings of the 32nd International Conference on Machine Learning (ICML-15)", vol. 37 (2015b).

Wang, J., J. Zhou, P. Wonka and J. Ye, "Lasso screening rules via dual polytope projection", in "Advances in Neural Information Processing Systems", pp. 1070–1078 (2013a).

Wang, Y., Y. Song, P. Rajagopalan, T. An, K. Liu, Y. Y. Chou, B. Gutman, A. W. Toga and P. M. Thompson, "Surface-based TBM boosts power to detect disease effects on the brain: an N=804 ADNI study", Neuroimage **56**, 4, 1993–2010 (2011b).

Wang, Y., Z. J. Xiang and P. J. Ramadge, "Lasso screening with a small regularization parameter", in "Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on", pp. 3342–3346 (IEEE, 2013b).

Weber, G. H., O. Rubel, M.-Y. Huang, A. H. DePace, C. C. Fowlkes, S. V. Keranen, C. L. Luengo Hendriks, H. Hagen, D. W. Knowles, J. Malik *et al.*, "Visual exploration of three-dimensional gene expression using physical views and linked abstract views", IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) **6**, 2, 296–309 (2009).

Xiang, S., L. Yuan, W. Fan, Y. Wang, P. M. Thompson, J. Ye, A. D. N. Initiative *et al.*, "Bi-level multi-source learning for heterogeneous block-wise missing data", NeuroImage **102**, 192–206 (2014).

Yang, T. *et al.*, "Detecting genetic risk factors for alzheimer's disease in whole genome sequence data via lasso screening", in "IEEE International Symposium on Biomedical Imaging", pp. 985–989 (2015).

Yuan, M. and Y. Lin, "Model selection and estimation in regression with grouped variables", Journal of the Royal Statistical Society: Series B **68**, 1, 49–67 (2006).

Zhang, D., D. Shen, A. D. N. Initiative *et al.*, "Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in alzheimer's disease", Neuroimage **59**, 2, 895–907 (2012).

Zhang, J., Y. Fan, Q. Li, P. M. Thompson, J. Ye and Y. Wang, "Applying sparse coding to surface multivariate tensor-based morphometry to predict future cognitive decline", in "Biomedical Imaging (ISBI), 2017 IEEE 14th International Symposium on", (IEEE, 2017a).

Zhang, J., Q. Li, R. J. Caselli, P. M. Thompson, J. Ye and Y. Wang, "Multi-source multi-target dictionary learning for prediction of cognitive decline", in "International Conference on Information Processing in Medical Imaging", (Springer, 2017b).

Zhang, J., J. Shi, C. Stonnington, Q. Li, B. A. Gutman, K. Chen, E. M. Reiman, R. Caselli, P. M. Thompson, J. Ye *et al.*, "Hyperbolic space sparse coding with its application on prediction of alzheimer's disease in mild cognitive impairment", in

"International Conference on Medical Image Computing and Computer-Assisted Intervention", pp. 326–334 (Springer, 2016a).

Zhang, J., C. Stonnington, Q. Li, J. Shi, R. J. Bauer, B. A. Gutman, K. Chen, E. M. Reiman, P. M. Thompson, J. Ye *et al.*, "Applying sparse coding to surface multivariate tensor-based morphometry to predict future cognitive decline", in "Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on", pp. 646–650 (IEEE, 2016b).

Zhang, J., Y. Wang, Q. Li, J. Shi, R. Bauer, K. Chen, E. M. Reiman, R. J. Caselli and C. M. Stonnington, "Patch-based sparse coding and multivariate surface morphometry for predicting amnestic mild cognitive impairment and alzheimer's disease in cognitively unimpaired individuals", Alzheimer's and Dementia: The Journal of the Alzheimer's Association **12**, 7, P947 (2016c).

Zhang, J., B. Yang and L. Jing, "Multiple similarity matrices for multi-label learning", JOURNAL OF INFORMATION and COMPUTATIONAL SCIENCE **11**, 16, 5843–5854 (2014).

Zhang, R. and J. T. Kwok, "Asynchronous distributed admm for consensus optimization.", in "Proceedings of the 31st International Conference on Machine Learning (ICML-14)", pp. 1701–1709 (2014).

Zhang, T., "Solving large scale linear prediction problems using stochastic gradient descent algorithms", in "Proceedings of the twenty-first international conference on Machine learning", p. 116 (ACM, 2004).

Zhou, J., J. Liu, V. A. Narayan and J. Ye, "Modeling disease progression via fused sparse group lasso", in "Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 1095–1103 (ACM, 2012).

Zhu, D. *et al.*, "Large-scale classification of major depressive disorder via distributed lasso", in "12th International Symposium on Medical Information Processing and Analysis", p. 10160 (International Society for Optics and Photonics, 2017).